1981

The effect of team programming on student achievement in COBOL instruction

Nancy Ellen Miller
Iowa State University

INFORMATION TO USERS

This was produced from a copy of a document sent to us for microfilming. While the most advanced technological means to photograph and reproduce this document have been used, the quality is heavily dependent upon the quality of the material submitted.

The following explanation of techniques is provided to help you understand markings or notations which may appear on this reproduction.

1. The sign or "target" for pages apparently lacking from the document photographed is "Missing Page(s)". If it was possible to obtain the missing page(s) or section, they are spliced into the film along with adjacent pages. This may have necessitated cutting through an image and duplicating adjacent pages to assure you of complete continuity.

2. When an image on the film is obliterated with a round black mark it is an indication that the film inspector noticed either blurred copy because of movement during exposure, or duplicate copy. Unless we meant to delete copyrighted materials that should not have been filmed, you will find a good image of the page in the adjacent frame. If copyrighted materials were deleted you will find a target note listing the pages in the adjacent frame.

3. When a map, drawing or chart, etc., is part of the material being photographed the photographer has followed a definite method in "sectioning" the material. It is customary to begin filming at the upper left hand corner of a large sheet and to continue from left to right in equal sections with small overlaps. If necessary, sectioning is continued again—beginning below the first row and continuing on until complete.

4. For any illustrations that cannot be reproduced satisfactorily by xerography, photographic prints can be purchased at additional cost and tipped into your xerographic copy. Requests can be made to our Dissertations Customer Services Department.

5. Some pages in any document may have indistinct print. In all cases we have filmed the best available copy.

University
Microfilms
International

300 N. ZEEB RD., ANN ARBOR, MI 48106

MILLER, NANCY ELLEN

THE EFFECT OF TEAM PROGRAMMING ON STUDENT ACHIEVEMENT IN COBOL INSTRUCTION

Iowa State University PH.D. 1981

PLEASE NOTE:

In all cases this material has been filmed in the best possible way from the available copy. Problems encountered with this document have been identified here with a check mark __√__.

1. Glossy photographs or pages _____
2. Colored illustrations, paper or print _____
3. Photographs with dark background _____
4. Illustrations are poor copy _____
5. Pages with black marks, not original copy _____
6. Print shows through as there is text on both sides of page _____
7. Indistinct, broken or small print on several pages ✓
8. Print exceeds margin requirements _____
9. Tightly bound copy with print lost in spine _____
10. Computer printout pages with indistinct print ✓
11. Page(s) _____ lacking when material received, and not available from school or author.
12. Page(s) _____ seem to be missing in numbering only as text follows.
13. Two pages numbered _____. Text follows.
14. Curling and wrinkled pages _____
15. Other _____

The effect of team programming on student achievement in COBOL instruction

by

Nancy Ellen Miller

A Dissertation Submitted to the
Graduate Faculty in Partial Fulfillment of the
Requirements for the Degree of
DOCTOR OF PHILOSOPHY

Department: Professional Studies in Education
Major: Education (Higher Education)

Approved:

Signature was redacted for privacy.

In Charge of Major Work

Signature was redacted for privacy.

For the Major Department

Signature was redacted for privacy.

For the Graduate College

Iowa State University
Ames, Iowa
1981

TABLE OF CONTENTS

CHAPTER I.

INTRODUCTION

As a result of analyzing the art of computer program development, new skills are being suggested to improve programming efficiency. Two of these skills are the use of structured programming and the ability to work as a member of a team (Khailany and Saxon, 1978). The benefits gained from employing these skills have been discussed in the literature (Cheney, 1977; Lemos, 1978; Weinberg, 1971) and in commercial areas (Baker and Mills, 1973; Schonberger and Franz, 1978).

Structured programming is a method of designing computer programs that provides readability, maintainability, and reliability. Readability is an important feature of good programming since it facilitates maintenance. Maintainability is concerned with reducing the time required to correct malfunctions or make design modifications. Maintainability is enhanced since structured programs are composed of modules of code that can be easily removed and replaced. Reliability is based on how long the program operates before it fails. Increased reliability in programs is obtained from reducing the number of component parts (modules), improving the reliability of each and using redundant modules.

Team programming involves the process of one's peers inspecting one's work for errors, ideas, and design methods in order to improve productivity and learning. The group

technique is being applied to programming in industrial settings and is advocated for educational settings as one approach to improving programming. As Gruenberger states, cooperation is vital to our industry; students in computing should "learn to depend on the cooperation of others and to offer their cooperation to others" (Gruenberger, 1964, p. 6). Cheney (1977) supports this by advocating that programming instructors need to provide the necessary environment in which the desired skill development can take place. But little has been done to develop these skills in the classroom in preparation for careers in programming. In fact, the classroom environment has discouraged the development of team approaches. In the normal computer programming classroom, many instructors stress that programming assignments be done individually. Collaboration is often viewed as cheating. The result is an environment in which most learning must be accomplished on an individual basis. "This is a restrictive and undesirable situation that inhibits the degree of learning that can take place" (Cheney, 1977).

Under this approach, many programming instructors have stressed individuality in student programming because this facilitated measurement of a student's performance. Many students are secretive about their programming knowledge because they realize that one way they can gain an advanced standing in the class is to meet more deadlines than their

classmates. The result is a learning environment in which the learner is isolated and restricted from developing desirable cooperation and communication skills. However, it appears to be an environment in which the learner develops independence and self-reliance.

Statement of the Problem

There appears to be a conflict between the classroom instructional strategies which have evolved to promote language learning and the strategies which are advocated by industrial computer managers for developing programming skills. Language learning has generally been considered an individual endeavor whereas programming may be best achieved as a group activity. This research is designed to determine if the use of team methods in solving programming assignments affects student achievement. Student achievement in this case is measured by scores on the final examination.

Objectives of the Study

The primary objective of the study is to investigate the effect of group programming as an instructional activity. More specifically, the study is to determine whether significant end-of-semester differences exist between two particular groups of students enrolled in an Introduction to COBOL Programming course at the University of Wisconsin - La Crosse, when a control group writes programs in the traditional

individualized manner and the experimental group writes programs in teams of three. Differences in COBOL language achievement will be measured along three dimensions:

1. knowledge of grammatical structure and syntax rules,
2. ability to read programs and determine the output of a program given its input and,
3. the ability to write a COBOL program or program segments, given a statement of the problem.

The secondary objective is to attempt to contribute much needed research to areas of teaching and learning a computer programming language.

Hypotheses to be Tested

There are no significant differences between students who complete class programming assignments on their own and students who complete class programming assignments in teams

1. in the scores on the end-of-term COBOL language grammar/syntax portion of the final exam
2. in scores on the end-of-term COBOL reading portion of the final exam
3. in scores on the end-of-term COBOL program writing portion of the final exam
4. in average programming scores
5. in course grade.

In addition to the formal hypotheses, descriptive data will be collected in an attempt to gain insight into student

acquisition of programming skills.

Basic Assumptions

The students' understanding of programming can be effectively ascertained by means of programming knowledge tests administered under time constraints.

Limitations of the Investigation

1. The students will not be randomly assigned to classes by the investigator.

2. The time of day during which courses are offered will not be controlled by the investigator.

3. The study will be limited to three instructors at a single school and to content of one course.

4. The study will compare only two approaches to programming instruction, namely individualized programming and team programming with randomly formed teams for each programming assignment.

5. The number of computer runs per programming assignment will not be available to the investigator for comparison.

6. The evaluation will be based on student performance in a beginning computer programming course.

Definition of Terms

COBOL is an acronym for COmmon Business Oriented Language. This is a common procedural language designed for commercial data processing as developed and defined by a national committee of computer manufacturers and users. The English-language statements of COBOL provide a relatively machine-independent method of expressing a business-oriented problem to the computer. Commonly used nouns, verbs, and connectives are used in the procedural portion of a COBOL program to construct easily understood sentences. The excellent documentation provided by COBOL - problem definition as well as a method of solution - enables more than one programmer to work on a particular problem with minimal duplication of effort (Sippl and Sippl, 1974, p. 74).

Structured programming is a standard methodology for creating programs that are elegant because of their simplicity, reliable because of the reduction of the number of "moving parts", maintainable because of their modular structure and minimally documented because they are so explicit. Structured Cobol eliminates the need for external documentation except for a simple structure chart and some minimal level of descriptive narrative (Jackson, 1980, p. 49-54).

Team programming is one method of programming consisting of "a collection of programmers who are trying to produce a single program by working together (Weinberg, 1971).

CHAPTER II.

REVIEW OF LITERATURE

Computer Programming

Millions of computers are in use in the world today and their numbers continue to increase despite the fluctuations of economic cycles. An interesting phenomenon is that during the lifetime of the computer we have had general economic inflation; however, the cost of computer hardware continues to decrease. The reason for the decrease in hardware costs is the continuing technological advancements made in the computer industry. Internal computer hardware components have become miniaturized, internal storage devices have increased 100 times in capacity, and data can be transferred at least 8 times faster than in early computers (Bohl, 1980). As a result of the technclogical advances, the power and capabilities of tne computer have continued to increase. Much of the sophistication of present computing systems is possible because of larger and more complex software systems that enhance tne hardware.

The need for research in the writing of software is demonstrated by the economic importance of software development. In 1973, the overall software cost in the U. S. was estimated to exceed $10 billion, surpassing hardware costs by a factor of two to one (Litecky and Davis, 1976, p. 33). Furthermore, Weinberg estimates that by 1985, software costs

will account for 90 percent of the total software and hardware computing costs (Weinberg, Wright, Kauffman, and Goetz, 1977). The economic importance of these statistics justifies an extensive research effort to improve software and to facilitate its development and maintenance. In no area is the need as great as in the use of COBOL, which according to the Philippakis survey supports 86% of the programming effort (Philippakis, 1973).

Numerous attempts have been made to improve the efficiency of program development. Various approaches to improving documentation, increasing programmer productivity, and increasing compiler efficiency have been tried with moderate success being reported (Ulloa, 1980; and Youngs, 1974). The most promising area for improvement currently appears to be in increasing the readability of programs. Dijkstra states his views on the need for program readability in Structured Programming:

> ... one of our aims is to make such well-structured programs that the intellectual effort ... needed to understand them is proportional to program length ... (Dahl, Dijkstra, and Hoare, 1972, p. 20).

The need for structured coding and improved readability is expressed by McGowan stating that

> ... human readability is a principal objective of structured coding. Accordingly it facilitates maintenance whose costs often exceed development costs... (McGowan, 1975, p. 25-26).

Since one of the properties of programs is that they will

probably have to be modified in the course of their lifetime, and since readability facilitates maintenance, according to McGowan (1975), improving the readability of programs should facilitate program maintenance. Program maintenance is often performed by personnel who are not the original authors and who often know little about the program to be maintained. Therefore, there has developed a need for easily modifiable programs and for "error free" programs in order to reduce the amount of maintenance to production programs (Lemos, 1979).

Even in program development, programmers spend more time debugging code than writing it; the problem being one of organization (Baker and Mills, 1973). One approach used by industry to help structure and organize the task of programming is the programmer teams. IBM has found that programmer team operations substantially improve quality and productivity of programming by reducing the debugging and reworking required in a product (Baker and Mills, 1973). The fact that two people work on a program as a team insures that at least two people understand the developing program. The programmer team method of organization is implemented most effectively when structured programming techniques are also practiced (Bohl, 1980). A number of IBM projects included the use of structured programming techniques by programmer teams. The productivity of these projects was significantly higher than previously experienced without structured programming and

programmer teams (Baker and Mills, 1973).

Baker (1972) found the team approach to be desirable in industry because the programming efficiency was substantially higher. An information bank system was developed by a team of three programmers using structured programming techniques. The quality of the completed programs were superior to conventionally produced programs in terms of the number and level of errors remaining and its ease of maintenance. Baker feels programmer teams would probably double normal productivity. With the increased use of programmer teams by industry there is a growing need for personnel who are trained to serve as team members.

"The concept of programming in groups is not a new one" (Cheney, 1977, p. 1). Weinberg defined the concept in 1971 when he proposed the removal of the programmer's ego from the programming process by restructuring the social environment and a restructuring of the value system of the programmers in that environment (Weinberg, 1971). This concept has become known as "egoless programming". By allowing the team members to critique each other's code and make suggestions for improvements, he proposed that programming in groups would eliminate the ego problem. Weinberg cites several advantages for using the group structure in industry: 1) program specifications are more often met, 2) variations in debugging time are reduced which provide more realistic estimates on the

amount of progress made, 3) the adaptability of programs is improved since at least two people are capable of understanding the program, 4) the entire work is less susceptible to being delayed if one of the team members is sick, 5) the possibility of eliminating the most obviously inefficient areas is increased, and 6) the reading of others programs is a good training technique which is likely to raise the general level of competence of the group. Weinberg states that "... many successful software firms are based on this type of interaction", and strongly advocates the incorporation of team activities in a classroom environment that emphasizes "egoless programming" (Weinberg, 1971, p. 58).

Research in the Classroom

Several studies have involved using team activities in the classroom other than computer programming classes. In a study designed to measure the effects on student achievement in a math class when working in teams of three members, Pearl (1967) used the pretest-posttest control group design and matched the students into the experimental and control groups. He administered two standardized achievement tests, the California Intermediate Arithmetic Test and the Stanford Advanced Arithmetic Test, at the beginning and end of the semester. When the mean change in scores for the two groups were analyzed, the experimental group had a significantly higher mean gain score than the control group.

The results of the Goldman study which compared the performance of individuals with that of two-member groups, showed that:

1. students improve significantly when working in pair-groups over working as individuals,
2. the group effect is greatest for low level students,
3. students working with partners at their same intellectual level do not improve over working with partners below their level, and
4. students working with partners above their level do improve significantly over working with partners at or below their level, but the extent that the partners are above does not matter (Goldman, 1965).

Laughlin and Johnson (1966), in a similar study, found pair-groups improved more than individuals on the second testing of the Concept Mastery Test. Their study was based on the complementary type task model, in which each individual is assumed to possess some resources that are unshared by the other members of the groups. The combination of these shared resources within the group gives it superiority over the performance of the same individuals working independently. Laughlin and Johnson, concluded that members of a group pool provide complementary information and hence the group can surpass its best individual.

A pervasive objection to individual versus group problem solving is that group superiority over individuals may merely reflect the probability that the group will contain at least one individual who is above the mean ability level of the individuals who work alone. Laughlin, McGlynn, Anderson, and Jacobson cast doubt on this idea in a study of concept attainment by individuals versus cooperative pairs. The students performed problem-solving tasks first individually then secondly as pair-groups. The study showed there were no consistently "better" individuals so the dominance of such "better" individuals in the pair conditions would not explain the actual obtained superiority of the pairs over the individuals. In Laughlin's words, "the concept attainment of the cooperative pairs involved facilitative problem-solving processes beyond the dominance of the better problem solver in each pair" (Laughlin, McGlynn, Anderson, and Jacobson, 1968, p. 415).

An early study by Margaret Shaw (1932) compared individuals and groups on problem solving tasks. The subjects were members of a class in social psychology at Columbia University. In the first half of the experiment, half of the students worked in five groups of four persons each and the other half worked as individuals. In the second half, the roles of subjects were reversed. Shaw found that in the first half of the experiment, 7.9% of the solutions turned in

by individuals were correct as compared with 53% of the solutions turned in by the groups. The time required, however, was greater for groups than for individuals. In the second half, Shaw reported the number of correct solutions was again in favor of groups (27% correct as compared with 5.7% correct by individuals). The average times, however, were shorter for groups on two of the three problems. Shaw's results indicated that groups produced more correct solutions.

A similar study was undertaken by Husband (1940) in a study contrasting individuals and groups in terms of the man-hours required to arrive at a solution and the quality of the solution. The problems included arithmetic problems, a jigsaw puzzle, and code deciphering. Subjects included 120 college students, 40 of whom worked alone and 80 in pairs. He found that pairs were significantly better on the deciphering task and the jigsaw puzzle, but there was no significant difference between pairs and individuals on the arithmetic problems.

Many other investigators have reported results that are consistent with those cited above. Taylor and Faust (1952) compared individuals with groups of two and four persons on a modified version of "twenty questions," and found that individuals required more time and questions to identify objects than did groups. Again, the groups were relatively more costly in terms of man-minutes (an average of 7.40 minutes

for two-person groups, 12.60 minutes for four-person groups, and 5.06 for individuals). Using a complex intellectual problem, Barnlund (1959) compared the performance of individuals working alone, under majority rule, and as members of discussion groups. Decisions made by the discussion groups were better than those made either by individuals or by majority rule. A study by Tuckman and Lorge (1962) also demonstrated that groups of five persons had a greater probability of producing good solutions than did individuals. Finally Davis and Restle (1963), using three puzzle problems, compared four-person groups with individuals. The proportion of solutions was greater for groups than for individuals on all three problems. There were no differences in overall time, although individuals required fewer man-hours for each solution. In many of these studies reviewed, the amount of time spent on solving the problem was a factor in the students' achievement.

> Implicit in the studies on group problem solving is the hypothesis that interaction contributes something to the group product that is more than the mere combination of individual products. This hypothesis suggests that groups members somehow exert an influence on their fellow members which leads to behavior that would not occur when members are alone. If this is true, then the effect should not be limited to problem solving, but should also appear in learning phenomena (Shaw, 1971, p. 67).

Perlmutter and De Montmollin (1952) compared individuals and groups on a nonsense-syllable learning task. Half of the subjects worked in three-person groups, rested fifteen

minutes, then worked individually. For the other half, this order was reversed. Even though there were no significant differences between the two sets of subjects, the groups learned more and learned faster than individuals.

Yuker (1955) demonstrated the same effects in a learning task involving prose materials. Individual students were read a story, after which they were asked to recall it individually, then as a group, and finally as individuals. The group performance was better than the initial individual recall in 38 of the 40 groups and better than the best initial recall in 29 of the 40 groups. Yuker suggested that groups will learn more than individuals on tasks 1) on which several persons can work without getting in one another's way, 2) which can be solved through the addition of individual contributions, and 3) in which the parts of the solution are at least partially independent.

The optimum size a group should be in conducting studies of group versus individual learning is a point of discussion among researchers. In a study by Laughlin, Kerr, Davis, Halff, and Marciniak, college students took Part I of the Terman Concept Mastery Test as individuals and then retook the same test in cooperative high-ability or low-ability groups of size two through five or as control individuals. The results showed there was a direct linear increase in performance of high-ability groups with increasing group size

from 1 to 3, no difference from 3 to 4, and then a further increase from group size 4 to 5; there were no differences in performance of low-ability groups as a function of group size except for groups of size 4 to 5 versus control individuals indicating a weak effect of increasing group size (Laughlin, Kerr, Davis, Halff, and Marciniak, 1975).

Ziller (1957) asked individuals and groups of two to six persons to perform two different tasks: estimate the number of dots on a card that was displayed for 5 seconds, and select the four facts (from a list of 15) that were most critical in determining the correct answer to a complex problem. On both tasks, there was a significant positive relationship between group size and quality of performance, and in both cases there was a tendency for this positive relationship to become weaker as more and more members were added to the group. Thus, on the dot-judging task, three-person groups were 74% more accurate than individuals who worked alone, but six-person groups were only 9% more accurate than three-person groups. On the other hand, three-person groups scored 51% higher than individuals, and six-person groups earned scores that were only 18% higher than those groups containing three members.

A number of studies have been conducted using programmer team activities in the classroom. Schonberger and Franz cite three advantages of team assignment in computer programming

classes:

> "1) Students learn from peers instead of relying on the limited availability of consultants (whose advice is often not good anyway) and the instructor, 2) larger, more realistic programs may be written since the task is broken up into student-sized modules, 3) symbiotic advantages accrue from a system that is at once a learning procedure and a modern tool of the field-based practitioner" (Schonberger and Franz, 1978, p.75).

Cheney (1977) conducted a study with students in introductory Computer Science courses to examine the proposition that a group structure would increase programming productivity in the classroom sense. The control group was given instructions to work on programming assignments alone. The experimental group was instructed to check each others code for errors after initially coding their own version of the program. An exam was given at the end of the course. Cheney found that the mean exam score of those who programmed in pairs was better at a highly significant level than those who programmed individually. He also felt that the interaction between members provided for greater learning by the team members during the school term than by students programming individually.

In a similar study, Lemos (1978) found students who worked in teams showed superior performance in writing programs and required fewer average runs to complete homework assignments. Lemos used a pretest-posttest control group design in a university level introductory COBOL programming

language class.

The studies discussed above dealt with team activities in the classroom and several advantages were cited. However, the studies reviewed have inherent design and measurement problems according to Borg and Gall's description of research. Borg and Gall describe experimental research as follows:

> "A typical experimental design in education involves the selection of a sample of subjects; random assignment of these subjects to experimental and control groups; the exposure of the experimental group to a treatment that is withheld from the control group; and finally, the evaluation of the two groups on the dependent variable, that is, the variable that you are attempting to change" (Borg and Gall, 1979, p. 33).

According to this description these studies were not ideal. Most of them were not based on an experimental research design, control groups were not always used, and posttest scores were sometimes used but without any allowances for initial differences. A researcher should select an experimental design that is appropriate for the research problem. Borg and Gall feel that control-group designs allow for a study to be carried out more satisfactorily than single group designs. Control-group designs involve the use of two groups; an experimental group which receives a pretest, the experimental treatment and a posttest, and a control group which receives the same pretest and posttest as the experimental group but does not receive the treatment. So the con-

trcl group is kept as identical to the experimental group as possible except for the exposure to the experimental treatment. Then if extraneous variables have brought about changes between the pretest and the posttest, these will be reflected in scores of the control group. Thus, only the posttest change of the experimental group that is above the change in the contrcl group can be attributed to the experimental treatment.

Another important feature of experimental design is the random assignment of subjects to the experimental and control groups (Borg and Gall, 1979). Often times in field research, random assignment of subjects is not possible but the experiment can still be internally and externally valid. An experiment in which random assignment of subject is not made is termed a nonequivalent design (Campbell and Stanley, 1963). One part of a nonequivalent design is an appropriate pretest. The pretest can help insure the design is internally valid by determining whether the experimental and control groups are initially equivalent even though they have not been formed by random assignment. To help lessen the initial differences between treatment groups that may arise due to nonrandom assignment, the researcher should assign classrooms randomly to be experimental or control groups. The analysis of covariance technique should be used to statistically attribute the subjects scores to the effect of the experimental

treatment rather than differences in initial scores of the pretest (Borg and Gall, 1979). Simple gain scores of individuals are less desirable because of the possibility that the change in pretest to posttest scores may be a product of regression toward the mean rather than the effect of the experimental treatment (Campbell and Stanley, 1963). The appropriate index of the effectiveness of a learning situation is the amount of change which has taken place (Bloom, 1963). Although change score analysis is often condemned, Kenny (1975) feels it is perhaps the most common way of analyzing nonequivalent control group design. Bloom (1963) feels that the best technique available for dealing with gains is the analysis of covariance.

Evaluation of Programs

The studies discussed earlier did not specify what form of evaluation was used when grading student programs. An important feature in any study of students in a computer programming course is the evaluation of student-written programs (Lemos, 1977). A review of the literature reveals that little research has been devoted to designing an objective grading system for computer programs. Weinberg and Schulman (1974) suggested ranking the programs on such characteristics as 1) the number of program statements used, 2) the number of hours in completing the assignment, 3) output clarity, and 4) program clarity. This would be manageable for small classes

but not for class sizes of 30 to 40 or larger. Perhaps for large classes the programs could be given a rating on Weinberg and Schulman's characteristics on a scale of 1 to 5 where a score of 3 would be the average expectation of the instructor. This would quantify the grading process and make it more objective.

Van Tassel (1978) presents some specific points under the topic of program style or readability that should be considered:

1. one statement per line,
2. meaningful data names,
3. use of indentation to show program structure,
4. a modular design of commands.

Clutterham (1970) used the following guidelines for grading programs; assigning points for each category:

1. correct answers,
2. program efficiency in terms of length,
3. correct termination of program.

For partially correct output, the percent of correct output is multipled by the total points possible. The program efficiency is determined by comparing the number of statements in the student's program with the number of statements deemed by the instructor as the minimum necessary for completion of the assignment. This minimum standard is divided by the length of the student's program and this ratio multiplied by the

total points possible for this category:

program length points = standard length / student length x total points.

These first two categories made up 75% of the program grade. The remaining 25% was for correct termination of the program. If the program did not terminate normally, 25% of the grade was lost.

Evaluating student programs could be compared with evaluating student compositions. The composition rating scale used by the Cleveland Heights High School (Judine, 1965). contained three categories with the following percentages:

1. content - 50%,
2. style - 30%,
3. conventions - 20%.

The category of content could be translated to output of computer programs. The style category is also very important in programming with the recent trend toward structured programming. The category of conventions in terms of English compositions referred to sentence structure. In translating this category to computer programs, the sentence structure is built into the language since there is a finite list of commands and rigid requirements as to the limited number of possible ways each command can be used to make a program valid and run properly. The total points for a program should be divided into a small percentage for use of a flowchart in

design and equal percentages for output clarity and program style since the program style is stressed in a programming language course using a structured language (Miller and Petersen, 1980). Mayer (1975) found that the use of flowcharts assist students with program composition, i.e. in organizing the program.

The category of output clarity should include points for the correct output (and a percentage of these points for partially correct output), appropriate labeling of the output, and normal termination of the program with the correct output being weighted the heaviest. Also, the category of program style should include identification of the author, and remarks at the beginning of the program describing the program (Miller and Petersen, 1980). In addition, the program style category should include use of meaningful data names, modularization and indentation to show the structure of the program listing (Kernighan and Plauger, 1974). Program and output readability are measures of the clarity with which the meaning and purpose of the program and output are conveyed to the user (Weinberg and Schulman, 1974). Brooks (1975) cited three suggestions for improving the readability of a program: 1) labels, declaration statements and symbolic names should convey as much meaning as possible to the reader, 2) space and format should be used as much as possible to improve readability and show subordination and testing, and 3) the

necessary prose documentation should be inserted into the program as paragraphs of comment. If these suggestions are used at the time of writing a program, the resultant program will be self-documented, thus allowing for easier maintenance and insuring that documentation is always available to the program user. The categories of output clarity and program style are important components of a programming language used so widely in business.

ACT Tests

The American College Testing program tests are used to predict the success of high school students in college. With four subject area tests, measuring accomplishment in English usage, Math usage, Social Studies reading, and Natural Sciences reading, ACT provides an indication of the student's level of educational development. The tests attempt to assess each students general educational development and ability to do college-level work (Remigius, 1979).

In a study conducted at Chesapeake College, Black (1969) found that high school grade point average and ACT English test scores predict fairly accurately both academic achievement and dropout potential. In a similar study of Jackson State College freshmen, Funches (1967) concluded that the ACT Composite score is a more reliable predictor of first term success than the high school grade point average.

Munday (1968) studied the ACT tests in relation to intelligence measures (Henmom-Nelson Test of Mental Ability, revised edition, Grades 13-17, and the Otis Mental Ability Tests, Gamma Test), Scholastic Aptitude Test (SAT), English and reading test scores, high school rank and study habits. He concluded that scores on intelligence tests correlated highly with ACT test scores but moderately with high school grades. Sawyer and Maxey (1979), in a study over a four year period, made the same conclusion that grade predictions based on ACT scores showed slightly greater stability over time than those based on grades only.

A number of measures can be used for predicting the success of students in college. SAT and ACT are considered equal in their predictive power (Passons, 1967). High school rank is widely used as a predictor of success in college, both by itself and in combination with standardized test scores such as ACT and SAT (Munday, 1968).

The recent controversy over the use of ACT scores for college admission has to do with the current trend of nationally declining test scores. In a discussion of this topic, Cameron and Crockett (1978) states:

> "It is clear that the ACT Assessment is as effective today as it was ten years ago in communicating the probability of academic performance on the part of students who are entering postsecondary education" (Cameron, and Crockett, p. 703).

Summary

The review of literature revealed that not much research has been done in the area of improving the learning of computer programming languages, particularly the use of team activities. The concept of team work has been used in industry but only in a few cases in educational settings. In studies dealing with individual versus group problem solving, the literature reviewed strongly supports the conclusion that groups produce more and better solutions to problems than do individuals, although the differences in overall time required for solution are not consistently better for either individuals or groups.

The nonequivalent control group design, which includes a pretest to determine initial differences, a posttest to determine achievement, and the analysis of covariance of the subjects' scores to determine the effect of the experimental treatment, was cited as an appropriate experimental research design. This design controls for the main effects of history, maturation, testing and instrumentation.

The literature concerning the evaluation of student-written computer programs was reviewed. The components that were found to be important in evaluating computer programs were degree of program readability and style, correct answers, efficiency, output clarity and a modular design.

The literature concerning analysis of student achievement in team programming environments was reviewed. The time spent on programming projects and the ability to work as a team member were found to be important factors in that analysis.

The review of literature concerning the prediction of students success in college revealed that high school rank, high school grade point average, ACT and SAT test scores were fairly accurate predictors.

CHAPTER III.

METHODOLOGY

This investigation was designed to study the effect of group programming on student achievement in an introductory computer programming language class. Student achievement is measured in the areas of knowledge of grammatical structure and syntax rules, the ability to read programs, and the ability to write programs. The study was to determine whether significant end-of-semester differences existed between a control group which wrote programs in the traditional manner and the experimental group which wrote programs in teams of three.

Sample

The subjects included in the study were students enrolled in the four sections of Introduction to COBOL Programming, Computer Science 222 (henceforth CPTS 222), at UWL during the Spring and Fall semesters of 1980. Each semester, one of the four sections was randomly chosen as the control group with the remaining three (3) sections comprising the experimental group. One instructor had two experimental sections and the control section while the other instructor had the remaining experimental section. Seventy (70) students received a grade for the course after the Spring semester and 99 students received a grade for the course after the Fall semester. A complete set of data was unavailable for a few

of the subjects in the two semesters for a number of reasons. Some students never came to class, others were absent when the instruments were administered, and some students dropped the course before completion. A total of 26 subjects who had taken the pretest dropped the course during the two semesters (10 for Spring and 16 for Fall). Those students who dropped the course during the semester were not included as subjects in the analysis of data.

About two-thirds of those enrolled were male. For the most part, the students enrolled were evenly divided between those in their sophomore and junior year. CPTS 222 is required of all computer science majors so the majority of the subjects declared themselves to be majoring in Computer Science. One fourth (1/4) of the students declared themselves to be majoring in Business.

Description of the Instruments

The review of literature led the investigator to conclude that the time spent working on computer programming assignments and the ability to work as a team member are important factors in the analysis of the students' academic performance in a team programming environment. Since the investigation involved students and classes from two different semesters, biographical makeup and previous coursework are important factors in determining background of the subjects involved. The investigator, therefore, chose five instruments

to be administered to all students. They are 1) a biographical questionnaire, 2) an estimate of time spent on programming questionnaire, 3) an evaluation of fellow team members' participation, 4) a computer program evaluation, and 5) a pretest.

The biographical questionnaire consists of 7 items. A sample of the questionnaire is in Appendix A. The 7 items are:

1. name
2. major
3. year in school
4. reason for taking the course
5. sex
6. age
7. previous coursework in Computer Science (courses with semester and year taken).

The time estimate consists of three items concerning completion of the computer programming assignments:

1. estimate of the time spent on design
2. estimate of the time spent on writing (coding)
3. estimate of the time spent on testing and debugging.

This instrument was administered to all students upon completion of each programming assignment. A sample of the questionnaire is in Appendix B.

The evaluation of fellow team members consists of four items on which the rater was asked to rank (on a scale of one to five) the fellow team members:

1. effort in design stage
2. effort in coding stage
3. effort in debugging stage
4. overall effort as a team member.

The instrument was administered to all teams upon completion of all the programming assignments. A sample of the questionnaire is in Appendix C.

The computer program evaluation consists of 13 items on which each student's program will be scored by the instructor:

1. flowchart
2. author's name on program listing
3. remarks at the beginning
4. comments throughout program
5. meaningful data names
6. indentation to show structure
7. modularity of design
8. correct output
9. appropriate labeling
10. correct termination of program
11. program length

12. output embellishments

13. exemplary program style and clarity.

A sample of the scoring instrument is included in Appendix D.

The pretest of 50 multiple choice questions is included in Appendix E. The questions, covering grammar/syntax (questions 1 - 25) and the reading of programs (questions 26 - 50) have been selected by the investigator from an item analysis of previous exams given in Introduction to COBOL Programming during the Fall semester of 1979.

The reliability of the pretest instrument was determined using the students responses from the Fall 1979 semester to those 50 questions included in the pretest instrument. The Kuder-Richardson Formula 20 reveals a reliability coefficient equal to 0.85.

The reliability of the computer program evaluation form (Appendix D) was determined by the investigator and a colleague (Miller and Petersen, 1980). The method of testing that hypothesis was to select a programming class, have several professors score the same programming assignment, test those scores for significant differences, and compute reliability.

The COBOL programming class was selected for the experiment. Three professors from the Computer Science Department were selected to use the instrument to score programs; the teacher of the class, another professor who had taught COBOL,

and a professor who had taught computer programming but not COBOL. Each professor graded the same 10 programs which were selected randomly from 79 student programs.

Each of the ten (10) programs were scored by each professor without knowledge of the other's scores. The raw scores for the ten (10) programs scored by each of the three (3) professors are presented in Table F1 with the averages (75.7, 80.8, 71.8) and standard deviations (12.1, 10.4, 12.1) for each professor respectively. All statistical information was computed through the use of Statistical Package for the Social Sciences (SPSS). The analysis of variance listed in Table F2 indicates there is a significant difference among the raw scores of the three (3) professors. The raw scores were then ranked on a scale of 1 to 10, correcting for ties by taking the average. The ranks were then used as data through SPSS and are listed in Table F1. The correlation matrix of the ranked and raw scores in Table F3 reveals that the correlations among the ranks and among the raw scores given by the three (3) professors are high. The interrater reliability coefficient (alpha = 0.890), also listed in Table F3, indicates that the reliability of the ranks among the three (3) professors is highly significant. The interrater reliability coefficient (alpha = 0.979) in Table F3 indicates the reliability of the raw scores is also highly significant. The authors (Miller and Petersen, 1980), therefore concluded

that the instrument used in scoring the programs is a reliable measure of computer programs. The high interrater reliability, and the intercorrelations support the fact that the instrument can be used by more than one rater and still obtain consistent results. The content of the instrument is a collection of items from the literature cited that experts in the field feel are important in measuring a computer program. Prior to use, the instrument was reviewed by five computer science faculty members to establish content validity. The conclusion is that the instrument is indeed a valid instrument for measuring computer programs.

Criterion Variables

Four dependent variables were used in the study. The score on the final examination for the course was the principal dependent variable used in measuring academic performance. A 50 item test was developed for use as a pretest. The areas covered by the test consisted of knowledge of grammatical structure and syntax rules, ability to read programs and determine the output of a program given its input. The final examination consisted of the 50 multiple choice questions which were used as the pretest as well as short answer questions concerning the ability to write a COBOL program or program segment, given a statement of the problem. The other three dependent variables were:

1. the average programming score

2. the differences between the final examination and the pretest, excluding the writing protion of the final examination

3. the course grade.

The programming scores were determined by the instructors using the computer program evaluation form (Appendix D). The course grade was determined by the instructors and was based on programming assignments, common hourly examinations, and the common final examination.

Collection of the Data

Students from all four sections met on the first meeting of each week for a mass lecture and met with an instructor in a small assembly two more times each week for discussion. The class sizes ranged from 20 to 30 with a total initial enrollment of 100 to 120.

The large lectures given to all students were used to present the material that would be discussed further in that week's small assemblies.

Each instructor participated in presenting the lecture material to all the CPTS 222 students in the Monday mass meeting in a large lecture hall. The lecture topic and the emphasis of each topic was governed by an outline that specified the order and amount of time intended for each topic. The discussions in the small assemblies for the next two meetings of the week followed the outline as well in terms of

the topic, emphasis and amount of time intended. The three common hourly examinations and common final examinations were used to assure that each instructor had covered the topics. The course met for 46 periods of 50 minutes each. The common examinations were given to all sections at the same time in a large lecture hall.

The same programming assignments were made for all sections. The programs assigned were all written in COBOL and specifically dealt with the following areas or features of COBOL: simple input and output, arithmetic calculations, intermediate totals, arrays, the SEARCH verb, and the SORT verb. The purpose of each program was to help the student focus attention on a particular COBOL feature. A week by week outline listing topics and programming assignments is in Appendix G.

The pretest instrument was administered to the students during the first regularly scheduled small assembly in all class sections of each semester. The biographical questionnaire was administered on the second regularly scheduled small assembly in all class sections of each semester. The common hourly exams were administered to the students during regularly scheduled large lectures in a large lecture hall. The common final examinations were administered to the students during the scheduled final examination period in a large lecture hall.

All instruments were administered by the instructors for the course. The pretest instrument required the entire 50 minute class period, the biographical questionnaire required about 10 minutes of class time, while the final examination required the entire 2 hour final examination period.

The programs assigned during the semester were scored by the individual instructors, using the computer program evaluation form (Appendix D). There were differences in the scoring of programs using the computer program evaluation form but the three (3) sections that comprised the experimental group were spread over two instructors to try to equalize this difference. The estimate of time spent and the evaluation of team member instruments were given to the students when the assignments were made and returned to the instructors when the program was completed.

The multiple choice questions on the pretest and posttest were machine scored while the writing portion was hand scored by the instructors for the course. The biographical questionnaire was hand scored by the investigator. The biographical information, the pretest score, the common hourly examination scores, the final examination scores, the programming scores, the estimates of time spent and evaluations of team members were recorded on punched cards for purposes of statistical analysis.

Description of Analysis

Data analysis programs were written to calculate the average time students spent on programming assignments, the average evaluation score given by a team member and the averageprogram score. Special values were coded for missing data in order to sort out the subjects on whom complete information was unavailable. The Statistical Package for the Social Sciences (SPSS) was used to perform an analysis of covariance using computer science grade point average as the covariate, and Pearson Product-Moment Correlation on the data.

The analysis of covariance subprogram prints the sum of squares, degrees of freedom, mean squares, F ratios and probabilities associated with each F ratio for the covariates and factor main effects. Then, factor main effects are assessed with adjustments made for other factors and all covariates.

Subprogram PEARSON CORR computes the Pearson product-moment correlation for pairs of variables. The Pearson correlation coefficient r is used to measure the strength of relationship between two interval-level variables. Output from this program includes the coefficient, the test of significance, and the number of cases upon which the correlation coefficient was computed. SPSS was used to determine whether significant end-of-semester differences existed between those

who wrote programs in an individualized manner and those who wrote programs in teams of three.

CHAPTER IV.

ANALYSIS OF DATA

Data are presented in tabular form for both Spring and Fall semesters. The analysis of the data was divided into several parts. First, relationships between biographical variables and the criterion variables were explored using the Pearson product moment correlation. Second, the biographical variables for the experimental and control groups and the Spring and Fall semesters were analyzed using Student's t test. The mean and standard deviation tables are given for all biographical variables and for all dependent variables (average programming score, the pretest and the posttest score on the grammar/syntax portion, the pretest and the posttest score on the reading portion, the score on the writing portion of the final exam, and course grade). Intercorrelation matrices are given for all independent variables and criterion variables.

The sample size for the posttest scores is smaller than the sample size for course grade because some students (10 total for both semesters) failed to take the final exam. By not taking the final exam they are assured of getting a grade of 'F' which makes them eligible to repeat the course.

The following abbreviations will be used in the tabulation of information.

year in college (year)

number of computer science hours completed in college (CS HRS)

computer science grade point average (CS GPA)

college grade point average (GPA)

English score on ACT (ENG ACT)

Mathematics score on ACT (MATH ACT)

Social Studies score on ACT (SOCS ACT)

Natural Science score on ACT (NATS ACT)

Composite score on ACT (COMP ACT)

sample size (N)

standard deviation (Std Dev or S.D.)

Student's t value (t)

Spring Semester 1980 (S80)

Fall Semester 1980 (F80)

degrees of freedom (d.f.)

average programming score (avgprog)

writing portion of final examination (postw)

syntax portion of pretest (pres)

reading portion of pretest (prer)

syntax portion of posttest (posts)

reading portion of posttest (postr)

course grade (grade)

experimental group (X)

control group (C)

pretest/posttest (test)

experimental/control (group)

Biographical

A summary of the means and standard deviations for all biographical variables for both Spring and Fall semesters is presented in Table I. No significant differences were found in any of the variables except year in college. The t test indicated a highly significant difference between semesters for the variable in year-in-college with the students in the Fall semester having completed more credits than the students in the Spring semester. Students in the Fall semester were approximately 1 semester ahead of the students in the Spring semester (2.18 for Spring, 2.71 for Fall).

The correlation coefficients between the criterion variables and the biographical variables are listed in Table II. College grade point average was found to have weak to moderate correlations with all portions of the posttest and grade for both semesters except the posttest writing for the Spring semester. Computer science grade point average was weakly to moderately correlated with all criterion variables except average programming score and the pretest syntax score

TABLE I
Means and Standard Deviations
for Biographical Variables

Biographical Variables	t	N	Mean S80	N	Mean F80	Standard Deviation S80	Standard Deviation F80
Year	-3.53**	80	2.18	115	2.71	1.04	1.05
Age	-0.73	80	21.15	115	21.61	3.94	4.56
CS HRS	-1.47	80	5.14	115	5.78	3.12	2.93
CS GPA	-1.46	80	2.76	115	2.96	1.13	0.82
GPA	-1.18	80	2.81	115	2.93	0.82	0.71
ENG ACT	-1.07	46	19.56	56	20.43	4.54	4.24
MATH ACT	0.59	46	25.59	55	25.04	3.85	5.30
SOCS ACT	-0.56	46	21.41	55	22.11	5.66	6.61
NATS ACT	0.23	46	26.57	55	26.35	4.31	5.19
COMP ACT	-0.10	46	23.50	55	23.58	3.30	4.32

** $p < 0.01$.

TABLE II
Correlations of Criterion Variables with Biographical Variables
S80
F80

Biographical Variables	Avgprog N=70 N=113	Pres N=68 N=115	Prer N=68 N=115	Posts N=68 N=91	Postr N=70 N=91	Postw N=68 N=91	Grade N=70 N=99
Year	.240*	.137	.167	.069	.106	.278*	.210
	-.097	-.060	.020	-.118	-.220*	-.074	-.016
Sex	.009	.122	-.098	-.257*	-.098	-.121	-.126
	-.127	.023	-.079	-.117	-.002	-.026	-.104
Age	.178	.067	.182	.083	.137	.238	.181
	.182	.067	-.044	.048	-.018	.131	.094
CS HRS	.023	.438**	.380**	.107	.069	.140	.210
	.048	.141	.223*	.104	.046	.108	.146
CS GPA	.056	-.154	.063	.222	.232	.074	.228
	-.006	.157	.269**	.306**	.441**	.591**	.473**
GPA	.214	-.018	.098	.268*	.410**	.211	.340**
	.226*	.119	.141	.310**	.357**	.457**	.536**

* $p < 0.05$.

** $p < 0.01$.

for Fall. The syntax score on the posttest had a weak correlation in a negative direction with sex for Spring indicating the female subjects tended to have higher scores than male subjects. The number of computer science hours was weakly correlated with the reading scores on the pretest for both semesters amd with the pretest syntax score for Spring. The year in college was weakly correlated with average programming score and writing portion of the posttest for the Spring semester.

The correlation coefficients between the criterion variables and the ACT scores are listed in Table III. The reading score on the pretest was weakly correlated with the Math and Composite ACT variables for both semesters. The writing score on the posttest was moderately correlated with all ACT variables for Fall only. The Math ACT variable was moderately correlated with grade for both semesters.

Correlation coefficients between all biographical variables are listed in Table IV. Moderate correlations were found between age and year in college, computer science grade point average with college grade point average, with English ACT scores for both semesters but with the Math, Social Studies and Composite ACT scores for Fall only. Moderate correlations were found among most of the ACT scores. College grade point average is weakly correlated with the Math, Social Studies and Composite ACT variables for Fall only.

TABLE III
Correlations of ACT Scores
with Criterion Variables
S80
F80

	Avgprog	Pres	Prer	Posts	Postr	Postw	Grade
ENG ACI	.032	-.359*	.059	.085	.115	.128	.126
	-.240	.110	.277*	.089	.197	.406**	.210
MATH ACT	.193	-.119	.330*	.329*	.324*	.108	.421**
	-.197	.070	.364**	.239	.304	.500**	.397**
SOCS ACT	.121	.018	.368*	-.018	-.098	.046	-.024
	.017	.054	.220	.030	.102	.532**	.228
NATS ACT	-.053	.001	.186	.005	.003	.207	.002
	-.034	.042	.242*	.166	.204	.520**	.238
COMP ACT	.115	-.110	.370*	.095	.067	.160	.175
	-.144	.133	.369**	.128	.207	.589**	.321*

* P < 0.05.

** p < 0.01.

TABLE IV
Correlation Matrix for Biographical Variables
S80
F80

Sex	-.135 -.006					
Age	.625** .446**	.158 -.083				
CS HRS	.152 .040	-.037 .020	-.006 .063			
CS GPA	-.109 .026	-.222* -.034	-.235* .173	.095 -.017		
GPA	.190 -.027	-.175 -.137	.015 .150	-.026 -.021	.625** .479**	
ENG ACT	.080 -.152	-.212 -.071	.176 -.041	-.269 -.066	.371* .427**	.285 -.023
MATH ACT	.077 -.156	-.051 .125	-.002 -.449**	-.033 .218	.280 .455**	.203 .265*
SOCS ACT	.206 .017	.143 .035	.274 -.075	-.145 .001	-.027 .319*	.012 .335*
NATS ACT	.042 .078	.038 .160	.037 -.218	.132 -.046	-.035 .229	.032 .138
COMP ACT	.201 -.083	-.021 .082	.189 -.282*	-.067 -.045	.172 .434**	.173 .341*
	Year	Sex	Age	CS HRS	CS GPA	GPA

* $p < 0.05$.

** $p < 0.01$.

TABLE IV (continued)

MATH ACT	.313* .321*			
SOCS ACT	.466** .691**	.165 .302*		
NATS ACT	.418** .548**	.116 .569**	.604** .718**	
COMP ACT	.717** .768**	.497** .671**	.838** .877**	.762** .888**
	ENG ACT	MATH ACT	SOCS ACT	NATS ACT

Moderate correlations in a negative direction were found for Fall in age with the Math and Composite ACT scores.

Criterion

Correlation coefficients between the criterion variables are presented in Table V. The average programming score was found to have weak correlations with course grade for both semesters. The syntax portion of the pretest was found to have a weak correlation with the reading portion of the pretest for Spring. The reading portion of the pretest was found to have moderate correlations with the writing portion of the posttest during the Spring semester but only weak correlations during the Fall semester. The reading portion of the posttest also had moderate correlations with the grade and posttest syntax for the Spring semester. Moderate to high correlations were found among all portions of the posttest and grade. The pretest syntax showed little or no correlation with the criterion variables.

Treatment

The means and standard deviations for both treatment groups (experimental and control) for all biographical variables for students from the Spring and Fall semesters are presented in Table VI. The t test indicated a significant difference between treatment groups in computer science grade point average and year in college for Fall with the control

TABLE V
Correlation Matrix for Criterion Variables
S80
F80

Pres	.174 .080					
Prer	.192 .152	.278* .164				
Posts	.173 .189	.120 .203	.330** .047			
Postr	.165 .149	.025 .183	.257* .149	.644** .670**		
Postw	.216 .173	.146 .160	.407** .206*	.642** .506**	.721** .539**	
Grade	.481** .288**	.172 .142	.438** .183	.685** .645**	.716** .633**	.788** .785**
	Avgprog	Pres	Prer	Posts	Postr	Postw

* $p < 0.05$.

** $p < 0.01$.

group being the higher of the two groups. The t test indicated a highly significant difference between treatment groups for Fall in age for Fall semester with the students in the control group being almost 3 years older. The ages of the students in the Spring semester, while not significantly different between groups, were on the average more than one year older in the experimental group. Significant differences were found between treatment groups for the Natural Science ACT variables for both semesters and the Composite ACT variable for Fall with the control group having the higher average score in all cases.

TABLE VI
Means and Standard Deviations
for Biographical Variables by Treatments
S80
F80

Biographical Variables	t	Mean N	X	N	C	Standard Deviation X	C
Year	1.22	56	2.27	24	1.96	1.07	0.96
	-2.22*	84	2.58	31	3.06	1.04	1.00
Age	1.20	56	21.70	24	20.40	4.28	3.52
	-2.80**	84	20.75	31	23.63	2.82	7.46
CS HRS	1.68	56	5.52	24	4.25	3.45	1.92
	-.84	84	5.64	31	6.16	2.90	3.02
CS GPA	-1.14	56	2.81	24	3.11	1.03	0.99
	-2.09*	84	2.91	31	3.29	0.84	0.76
GPA	-.39	56	2.81	24	2.89	0.78	0.93
	-.78	84	2.92	31	3.05	0.77	0.63
ENG ACT	-.21	32	19.41	14	19.71	4.46	4.87
	-1.73	46	19.98	10	22.50	4.39	2.80
MATH ACT	-.81	32	25.28	14	26.29	4.20	2.89
	-1.10	45	24.67	10	26.70	5.30	5.23
SOCS ACT	-1.39	32	20.66	14	23.14	5.89	4.83
	-1.72	45	21.40	10	25.30	6.45	6.67
NATS ACT	-2.35*	32	25.63	14	28.71	4.30	3.60
	-2.13*	45	25.67	10	29.40	5.16	4.30
COMP ACT	-1.54	32	23.00	14	24.64	3.39	3.18
	-2.19*	45	23.00	10	26.20	4.24	3.85

* $p < 0.05$.

** $p < 0.01$.

CHAPTER V.

FINDINGS

Four statistical techniques were used to analyze the data from the students participating in this investigation. Analysis of covariance was computed for each criterion variable using a treatment group by semester by test design with computer science grade point average as a covariate. Other available data concerning ACT scores, dropouts, estimate of time spent programming and evaluation of team members were analyzed using Student's t test, Pearson's product-moment correlation and analysis of variance.

Analysis of Covariance

The major concern of this investigation was to compare student achievement in COBOL instruction classes where the experimental group completed the programming assignments in teams of three and the control group completed the programming assignments individually. The computer science grade point average was used as a covariate since, as presented earlier, weak to moderate correlations were found with all of the criterion variables except average programming score. The significant or highly significant differences found in the analysis of covariance tests indicate that the covariate accounts for a significant amount of the variance in all of the criterion variables except average programming score.

Through the analysis of covariance, using treatment group (experimental and control), semester (Spring and Fall), and test (pretest and posttest) as independent variables, the explicit hypotheses were tested to determine the effect of team programming on the students' knowledge of syntax, reading ability, writing ability, average programming score, and grade. These analyses are discussed in turn below. In addition to these hypotheses, implicit hypotheses involving test and semester as well as the two- and three-way interactions were tested. The results of these analyses were also included in the tables.

> *Hypothesis 1:* There is no significant difference between the students in the control group and the students in the experimental group in the scores of the end-of-term COBOL language grammar/syntax portion of the final exam.

The results of this analysis are reported in table VII. During the Spring semester, the mean syntax score for the control group increased from 12.52 on the pretest to 34.67 on the posttest. The experimental group increased from 15.38 to 33.01. Fall semester's scores followed a similar pattern with the control group increasing from 13.85 to 35.65 and the experimental group from 13.88 to 33.94. Based on this analysis, this hypothesis was not rejected, ($F(1,319)=0.007$, ns). No significant differences were found between the treatment and control groups for syntax scores.

TABLE VII
Analysis of Covariance for Syntax
by Group, Test, and Semester
with Computer Science Grade Point Average as Covariate

Source of Variation	Df	Mean Square	F
CS GPA	1	207.644	4.752*
Group	1	9.017	0.206
Test	1	26177.891	599.122**
Semester	1	0.070	0.002
Group X Test	1	199.132	4.557*
Group X Semester	1	31.179	0.714
Test X Semester	1	51.298	1.174
Group X Test X Semester	1	13.535	0.310
Error	319	43.694	

* $p < 0.05$.

** $p < 0.01$.

TABLE VIII
Means and Standard Deviations
for the Syntax and Reading Scores
Adjusted by Computer Science Grade Point Average
S80
F80

	Experimental Mean (S.D.)	Control Mean (S.D.)
Syntax		
Pretest	15.38 (5.73)	12.52 (5.24)
	13.88 (4.30)	13.85 (4.49)
Posttest	33.01 (8.87)	34.67 (8.93)
	33.94 (7.38)	35.65 (7.34)
Reading		
Pretest	13.72 (5.54)	12.82 (6.04)
	13.46 (5.30)	13.55 (5.88)
Posttest	36.50 (7.23)	37.97 (8.04)
	38.64 (6.54)	38.38 (7.12)

However, a significant interaction was found ($F(1,319)=4.557$, $p < .05$) between group and test. The pretest to posttest change was greater for the control group than for the experimental group. The control group gained approximately 15% more than the experimental group (22.15 points vs 17.63, respectively for Spring semester and 21.80 points vs 20.06, respectively for Fall semester).

No significant differences were found between the two semesters nor were the interactions between semester and group, or among test, group, and semester significant. Not surprisingly, the F ratio indicated a highly significant difference ($F(1,319)=599.122$), $p < .01$) between the pretest and posttest with the posttest scores being higher. The means and standard deviations for syntax scores (pretest and posttest) for experimental and control groups for both semesters are reported in Table VIII.

Using a similar analysis (Table IX), the following hypothesis was tested to determine the effect of team programming on the students' reading ability:

> Hypothesis 2: There is no significant difference between the students in the control group and the students in the experimental group in the scores of the end-of-term COBOL reading portion of the final exam.

Based on this analysis, this hypothesis was not rejected, ($F(1,319)=0.206$, ns). The F ratios indicate no significant difference between treatment and control groups

TABLE IX
Analysis of Covariance for Reading
by Group, Test, and Semester
with Computer Science Grade Point Average as Covariate

Source of Variation	Df	Mean Square	F
CS GPA	1	616.638	16.083**
Group	1	0.262	0.007
Test	1	37321.906	973.408**
Semester	1	11.407	0.298
Group X Test	1	58.427	1.524
Group X Semester	1	0.285	0.007
Test X Semester	1	42.088	1.098
Group X Test X Semester	1	16.316	0.426
Error	319	38.341	

** $p < 0.01$.

on reading scores. The control group made the greater change during the Spring semester increasing from 12.82 on the pretest to 37.97 on the posttest. The experimental group increased from 13.72 to 36.50. Fall semester's scores favored the experimental group with the greater change increasing from 13.46 to 38.64 and the control group from 13.55 to 38.38. No significant differences were found between the two semesters nor for the two- or three-way interactions. Like

the syntax scores, highly significant differences were found between the pretest and posttest. The means and standard deviations for reading scores are reported in Table VIII.

The third analysis used group (experimental and control) and semester (Spring and Fall) as independent variables to test the following hypothesis to determine the effect of team programming on the students' writing ability:

> Hypothesis 3: There is no significant difference between the students in the control group and the students in the experimental group in the scores of the end-of-term CCBCL writing portion of the final exam.

Based on the analysis, this hypothesis was not rejected ($F(1,154)=1.994$, ns). As shown in Table X, none of the main effects were significant. For the Spring semester data, the control group had a higher mean score (69.34) than the experimental group (65.79). Similarly for the Fall semester, the control group scored 69.94 versus 64.43 for the experimental group. No significant interactions were found. Means and standard deviations for the writing scores are reported in Table XI.

Since pretests on average programming score were not possible, average programming score was analyzed by using a treatment group by semester analysis of covariance with computer science grade point average as a covariate. The fourth analysis tested the following hypothesis:

> Hypothesis 4: There is no significant difference between the students in the control group and the

TABLE X
Analysis of Covariance for Writing
by Group and Semester
with Computer Science Grade Point Average as Covariate

Source of Variation	Df	Mean Square	F
CS GPA	1	7725.891	20.219**
Group	1	761.863	1.994
Semester	1	4.998	0.013
Group X Semester	1	372.081	0.974
Error	154	382.109	

** $p < 0.01$.

TABLE XI
Means and Standard Deviations
for the Average Programming Score,
Writing Portion of the Posttest, and Grade
Adjusted by Computer Science Grade Point Average
S80
F80

	Experimental Mean(S.D.)	Control Mean(S.D.)
Posttest		
Writing	65.79(19.16) 64.43(21.44)	69.34(19.95) 69.94(22.36)
Avgprog	88.13(9.38) 81.74(5.96)	79.95(20.16) 78.07(7.96)
Grade	2.62(0.97) 2.51(1.12)	2.34(1.39) 2.32(1.37)

> students in the experimental group in the average programming score.

The analysis in Table XII revealed that this hypothesis was rejected ($F(1,164)=11.092$, $p < .01$). Highly significant differences were found between treatment and control group with the treatment group having the higher average score. The average programming score for the experimental group for the Spring semester was 88.13 versus 79.95 for the control group. The Fall semester's scores followed a similar pattern with the experimental group averaging 81.74 for programs and the control group averaging 78.07. A significant difference ($F(1,164)=5.303$, $p < .05$) was also found between semesters which may be due in part to instructor differences. No significant interactions were found. The means and standard deviations for this variable are reported in Table XI.

A similar analysis was used to test the following hypothesis concerning the effect on grade:

> Hypothesis 5: There is no significant difference between the students in the control group and the students in the experimental group in the course grade.

This hypothesis was not rejected ($F(1,164)=1.405$, ns). The analysis, shown in Table XIII, reveals no significant differences between the treatment and control groups for the main effects. For the Spring semester, the average grade given the experimental group was 2.62 and 2.34 for the control group. Similarly for the Fall semester, the average

TABLE XII
Analysis of Covariance for Avgprog
by Group and Semester
with Computer Science Grade Point Average as Covariate

Source of Variation	Df	Mean Square	F
CS GPA	1	220.125	2.259
Group	1	1080.818	11.092**
Semester	1	516.743	5.303*
Group X Semester	1	148.467	1.524
Error	164	97.443	

* $p < 0.05$.

** $p < 0.01$.

TABLE XIII
Analysis of Covariance for Grade
by Group and Semester
with Computer Science Grade Point Average as Covariate

Source of Variation	Df	Mean Square	F
CS GPA	1	29.962	25.679**
Group	1	1.640	1.405
Semester	1	0.223	0.191
Group X Semester	1	0.464	0.398
Error	164	1.167	

** $p < 0.01$.

grade for the experimental group was 2.51 and for the control group 2.32. The means and standard deviations are reported in Table XI.

ACT Scores

ACT scores were available for the subjects and were analyzed to explore relationships between ACT|* SCORES AND ALL BIOGRAPHICAL AND CRITERION VARIABLES. THOSE SUBJECTS WHOSE ACT scores were unavailable were included in the analysis of data since all other data were complete. These subjects' ACT scores were unavailable either because their ACT scores were not reported when they were admitted to college, or they did not take the ACT test.

The means and standard deviations for the biographical variables of subjects with and without ACT scores available are presented in Table XIV. The t test indicates significant differences between the subject groups in year in college and age for both the Spring and Fall semesters. The significant differences in age may indicate that the older students did not have the ACT exam available for them, or the period of time between the time they took the test and the time they entered college was greater than for those subjects with ACT scores available.

The means and standard deviations for the criterion variables of subjects with and without ACT scores available are presented in Table XV. No significant differences were found

TABLE XIV
Means and Standard Deviations
for Biographical Variables of Subjects
with and without ACT Scores Available
S80
F80

Biographical Variables	Subjects without ACT Scores Mean	S.D.	N	Subjects with ACT Scores Mean	S.D.	N	t
Year	2.74	1.21	34	1.76	0.64	46	4.65**
	2.98	1.11	59	2.43	0.91	56	2.92**
Age	22.92	4.56	34	19.85	2.80	46	3.71**
	23.25	5.68	59	19.88	1.77	56	4.26**
CS HRS	5.35	3.82	34	4.98	2.52	46	0.53
	5.81	3.37	59	5.75	2.41	56	-0.12
CS GPA	2.71	1.18	34	2.79	1.09	46	-0.31
	2.99	0.89	59	2.94	0.75	56	0.30
GPA	2.90	0.72	34	2.74	0.88	46	0.89
	2.86	0.63	59	3.02	0.79	56	-1.21

** $p < 0.01$.

TABLE XV
Means and Standard Deviations
For Criterion Variables of Subjects
with and without ACT Scores Available
S80
F80

Criterion Variables	Subjects without ACT Scores Mean	S.D.	N	Subjects with ACT Scores Mean	S.D.	N	t
Avgprog	86.86	11.66	33	84.96	14.64	41	0.61
	80.26	7.60	59	81.41	5.83	54	-0.89
Pretest							
Syntax	14.59	4.96	34	14.48	6.29	46	0.08
	14.37	4.24	59	13.34	4.40	56	1.28
Reading	12.59	4.69	34	14.09	6.17	46	-1.17
	13.19	5.84	59	13.79	5.04	56	-0.59
Posttest							
Syntax	32.33	9.98	30	34.37	7.92	38	-0.94
	34.33	7.15	48	34.42	7.81	43	-0.05
Reading	36.40	6.98	30	37.32	7.88	38	-0.50
	38.00	6.56	28	37.21	6.81	04	-0.86
Writing	66.57	20.29	30	66.95	18.79	38	-0.08
	63.71	22.79	48	68.19	21.43	43	-0.96
Grade	2.55	1.12	31	2.54	1.10	39	0.04
	2.35	1.19	52	2.60	1.19	47	-1.04

between the subject groups for all criterion variables.

The results of this study seemed to indicate that Math and English ACT scores had moderate correlations with computer science grade point average. The Math ACT scores were also moderately correlated with the grade. From the results reported in this study, it appeared that Math and English ACT scores may provide predictive evidence as to how well a student will perform in a beginning computer programming language course.

Dropouts

Those subjects who started the course but did not finish it were called dropouts. It was not assumed that these subjects dropped the course only because they were receiving poor or failing marks during the first part of the course. Because of this assumption, the dropouts were not included as failing grades in the analysis for the criterion variables. There were 26 students who dropped the course for whom complete biographical data and pretest scores were available (10 from Spring and 16 from Fall).

The means and standard deviations for the important biographical variables and the pretest scores for dropouts and subjects finishing the course are presented in Table XVI. The results indicated that dropouts were generally less successful students than students completing the course. The dropouts had significantly lower computer science grade point

TABLE XVI
Means and Standard Deviations
for Important Biographical and Criterion Variables
by Dropouts and Subjects Finishing Ccourse
X
C

Variables	Dropouts Mean	S.D.	N	Subjects Finishing Mean	S.D.	N	t
Year	2.33	1.19	18	2.48	1.05	122	-0.58
	2.38	1.19	8	2.62	1.11	47	-0.56
CS GPA	2.29	1.07	18	2.86	0.92	122	-2.43**
	2.50	1.20	8	3.21	0.86	47	-2.04*
GPA	2.87	0.57	18	2.87	0.78	122	0.01
	2.28	1.16	8	2.98	0.77	47	-2.22*
Pres	13.98	4.09	18	14.53	5.09	122	-0.51
	11.50	4.87	8	13.66	4.83	47	-1.17
Prer	10.22	3.93	18	13.92	5.41	122	-2.79**
	12.75	5.55	8	13.66	6.03	47	-0.40

* $p < 0.05$.

** $p < 0.01$.

averages and pretest reading scores.

To determine if dropouts in the experimental and control groups differed in background or ability, comparisons were made using the analysis of variance. The F ratios for the analysis of variance are reported in Table XVII. There were no significant differences in dropouts between the treatment and control group for the variables listed. The data imply that the students did not drop the course because of the

TABLE XVII
F ratios from Analysis of Variance for Dropouts for Important Biographical and Criterion Variables by Treatment Group

Variables	Experimental Mean	S.D.	N	Control Mean	S.D.	N	F
Year	2.33	1.19	18	2.38	1.19	8	0.01
CS GPA	2.29	1.07	18	2.50	1.20	8	0.21
GPA	2.87	0.57	18	2.28	1.16	8	3.17
Pres	13.98	4.09	18	11.50	4.87	8	1.69
Prer	10.22	3.93	18	12.75	5.55	8	1.78

* $p < 0.05$.

** $p < 0.01$.

treatment but for other reasons. There were 13% drops in the experimental group and 15% in the control group.

Estimate of Time

The means and standard deviations of the estimates of time the students spent on designing, coding and debugging the programming assignments are presented in Table XVIII. The t test indicated a significant difference between treatment groups on the amount of time spent coding the programming assignments. The experimental group spent less time coding the programming assignments than the control group.

TABLE XVIII
t Test for Estimate of Time Variables by Treatment

Estimate of Time	Group	S80 N=70 Mean	S.D.	t	F80 N=99 Mean	S.D.	t
Design	X	2.31	2.28	0.18	3.30	3.39	0.01
	C	2.22	1.25		3.29	1.88	
Code	X	1.64	0.61	-2.55*	2.48	1.57	-2.03*
	C	2.17	1.08		3.16	1.22	
Debug	X	4.20	2.37	0.96	5.82	3.07	-0.61
	C	3.62	2.08		6.30	4.27	

* $p < 0.05$.

The correlations of the estimates of time spent programming with the average programming score are presented in Table XIX. Average programming score correlated negatively with the time spent designing, coding and debugging in both the Spring and Fall semesters. The results suggested that the teams approach led to superior programs being written with less time involvement for the students. The negative correlations for estimate of time spent on designing and debugging would seem to indicate that the control group spent more time on designing and debugging the programming assignments than the experimental group.

TABLE XIX
Correlations of Estimates of Time
with Average Programming Score

Estimate of Time Variable	X C	S80 N=50 N=20	F80 N=72 N=27
Design		-.362** .225	-.256* .194
Code		.203 .401*	-.243* .202
Debug		-.167 -.181	-.001 -.352*

* $p < 0.05$.

** $p < 0.01$.

Evaluation of Team Members

The correlations of the evaluations of team members with average programming score are presented in Table XX. The average programming score was weakly correlated with all evaluation items for Spring, Fall semesters, and the combined semesters scores. Such correlations between average programming score and the evaluation items may lend some viability to the idea of using the evaluation of team member form (Appendix C) as additional criteria in determining course grades for computer programming courses.

TABLE XX
Correlations of Evaluations of Team Members with Average Programming Score

Evaluation Variable	S80 N=50	F80 N=72
Design	.284*	.259*
Code	.284*	.286**
Debug	.274*	.236*
Overall	.298*	.239*

* $p < 0.05$.

** $p < 0.01$.

Test of Hypotheses

Hypothesis: There is no significant difference between the students in the control group and the students in the experimental group in the scores of the end-of-term COBOL language grammar/syntax portion of the final exam.

Data collected from the subjects during the Spring and Fall semesters, when statistically analyzed, provides no evidence on which to reject the hypothesis at the 0.05 level of significance for the grammar/syntax portion of the final exam. Although the subjects in the control groups consistently scored higher on the grammar/syntax portion of the final exam, it was not significantly so.

Hypothesis: There is no significant difference between the students in the control group and the students in the experimental group in the scores of the end-of-term COBOL reading portion of the final exam.

Data collected from the subjects during the Spring and Fall semesters, when statistically analyzed, projects no evidence on which to reject the hypothesis at the 0.05 level of significance for the reading portion of the final exam.

Hypothesis: There is no significant difference between the students in the control group and the students in the experimental group in the scores of the end-of-term COBOL writing portion of the final exam.

Data collected from the subjects during the Spring and Fall semesters provides no evidence on which to reject the hypothesis at the 0.05 level of significance for the program writing portion of the final exam. The control group consistently had the higher score of the two groups.

Hypothesis: There is no significant difference between the students in the control group and the students in the experimental group in the average programming scores.

Data collected from the subjects during the Spring and Fall semesters caused the hypothesis to be rejected at the 0.01 level of significance for the average programming scores. The experimental groups consistently having the higher scores.

Hypothesis: There is no significant difference between the students in the control group and the students in the experimental group in the course grade.

Data collected from the subjects during the Spring and Fall semesters, when statistically analyzed, provides no evidence on which to reject the hypothesis at the 0.05 level of significance for the course grade.

CHAPTER VI.

DISCUSSION

Results

The research was conducted to determine the effect of team programming on student achievement in a beginning computer programming language class. The results of this investigation indicated some very interesting patterns concerning student achievement in the areas of learning the syntax of a new computer language and the development of reading and writing abilities.

As was expected, the average programming scores for those subjects working in teams were better than for those subjects working alone. The programs were graded by the instructors without knowledge of whether the program was from the control group or the experimental group. These findings supported the research of Baker and Mills (1973) and Lemos (1978). These results also agree with the findings of Laughlin and Johnson (1966) that members of a team provide complementary information.

The fact that the teams had higher average programming scores indicated the students involved in team programming activities in a classroom situation were able to work together in some manner to produce a satisfactory program. This was of some initial concern to the researcher since the experimental group was under additional constraints of

scheduling work time together. Even under these constraints, they were more productive.

Some factors that could have affected the results of this study can be identified. One such factor was time. A significant difference found between the treatment and control groups on the amount of time spent coding the programming assignments. This supported the findings of Baker and Mills (1973) that programmer teams spend less time producing the better programs. Thus, the experimental group had more time to put in the extras to qualify for additional points in the scoring categories of program length, output embellishments, and program design. The control group, on the other hand, spent more time meeting the minimal requirements and may not have had time for the extras.

A second factor was that the programming activities were not monitored. It is not known if the teams actually worked together in all cases or if the members of the control group received outside help with the programming activities. A structured laboratory period for solving the programming projects may have provided some control on these situations.

A significant difference was found between semesters in average programming score. This difference may have been due in part to different instructors for the two semesters. The investigator taught two of the experimental sections and the control section and another instructor taught the third ex-

perimental section during the Spring semester. The differences found between instructors were minimal and were equalized somewhat by the division of sections between instructors. The investigator tended to grade the programs higher than the other instructor. This difference in grading would have a tendency to minimize the difference between the experimental and control groups. During the Fall semester, the investigator taught only one experimental section and another instructor (different from the Spring semester) taught the other two experimental sections and the control section. This instructor tended to grade lower than the investigator. The result would be a tendency to magnify the difference between the experimental and control groups.

There were no significant differences found between the experimental and control groups in their syntax, reading, or writing scores or in their grades. This is surprising since the investigator expected higher syntax scores for the control group. The significant syntax test-group interaction indicated that students in the control group improved more than did students in the experimental group. However, since the control group also was superior in intellectual ability, the results were difficult to interpret. One possibility was that students working as individuals do learn more syntax. A second possibility was that brighter students can be expected to gain more than less intelligent students. A third possi-

bility, given the large number of statistical tests performed was that a Type I error occured. The analysis of covariance used deleted part of the variance due to computer science grade point average but not all the variance. For those considering using team programming in classroon situations, consideration might be given to the use of computer assisted instruction or other supplementary instruction for the experimental group to help them with the syntax.

The lack of measureable main effects differences may be attributed in part to the measuring instruments. The syntax test consisted of questions which required students to select the COBOL statements which were correct in format. The reading test consisted of questions which required students simply to trace a small section of a COBOL program when given the input. A better measure of reading ability might have been to require the students to analyze an entire program without knowledge of input to determine the result of the program in general terms. It is possible that the exchange of approaches to program design within the experimental group would facilitate this activity.

Likewise, the writing portion of the exam measured the students' ability to write only small portions of a program. This activity tested the students' ability to choose the correct command to perform the required function and write the command correctly. This type of test might favor the

control group since those students who programmed individually had completely designed all program assignments. On the other hand, it might favor the experimental group since those students had the added benefit of being exposed to two other designs for each programming assignment. In either event, the creation of an entire program would require more ability to organize and design programs.

Analysis of the ACT scores resulted in moderate correlations between Math and English ACT scores with computer science grade point average and between Math ACT scores and grade in the course. The implication is that the Math and English ACT scores may have value in predicting student success in a beginning computer programming language course. High school rank might have been a better indicator than ACT scores but rank was not available to the investigator.

Conclusions

Several benefits can be identified in using team programming in classroom situations. First for the faculty involved, the team programming results in fewer programs to grade (one-third as many programs when using teams of three). Another benefit to the faculty, as stated by Schonberger and Franz (1978), is that students learn from their team members instead of asking the instructor or consultants for assistance.

Secondly, the computer center would benefit from team activities in the classroom. Fewer resources would be necessary and, as found by Lemos (1978), fewer runs would be made for each programming assignment.

Team programming, when used in upper division computer science courses would allow the students to complete larger more realistic projects in the course of a semester than could be done individually. This was one advantage to team assignments in computer programming classes cited by Schonbetger and Franz (1978).

Potential benefits not measured in this study are that of working together in a team situation. The measures used are traditionally established for individual learning. There were no affective measures to determine attitudinal changes that may have taken place as a result of working in teams.

Further Study

Several areas of concern were uncovered as a result of this investigation that need additional research. The first of these areas concerns the use of team programming at different levels of courses. This study dealt with a beginning computer programming course. Perhaps the results would be different in a computer programming course where the students use a higher level pseudo language that had a less detailed syntax. The students might be able to learn the structure and organization of programming first then move on to more

advanced courses to learn the syntax of a particular language and to apply the organization learned previously.

The second area concerns the potential benefits to industry. Baker and Mills (1973) indicated that industry was interested in improving the quality and productivity of programming through the use of structured programming and programmer teams. The use of programmer teams in the classroom in this study did not significantly detract from the students' achievement. It seems a follow-up study is needed to determine if the needs of industry were met by this approach.

A third area of concern is in the use of the computer program grading form. Differences were found in programming scores between instructors in the areas of "refinements above the minimum". This is a subjective area of the evaluation that scores the students on 1) efficiency of the program (program length), 2) usefulness of extra output (output embellishments), and 3) the approach to program design used (exemplary program style and clarity). These are important areas when considering programming skills. A better measure needs to be developed to make these categories more objective and possibly have more weight toward the programming score.

Finally, the effects of team programming on the achievement of students of different ability levels should be

investigated. The composition of the groups based on programming ability might moderate the effect of team programming.

CHAPTER VII.

SUMMARY

As a result of analyzing the art of computer program development, new skills are being suggested to improving programming efficiency. One of these skills is the use of team programming. The research was conducted to determine if the use of team methods in solving programming assignments effects student achievement. The subjects of this study were students enrolled in an introductory COBOL programming course at the University of Wisconsin - La Crosse during the Spring and Fall semesters of 1980. The subjects were divided into a control group who wrote programs in the traditional individualized manner and an experimental group who wrote programs in teams of three. Student achievement was measured in the areas of knowledge of grammatical structure and syntax rules, the ability to read programs and the ability to write programs. Data collected from the students included the score on the pretest, the average programming score, the score on the final exam and the course grade.

Results of the study indicate that those students involved in team programming had significantly better programming scores than the control group, showed moderately lower achievement in the areas of knowledge of grammatical structure and syntax, but showed no difference in achievement in the ability to write programs, or in the ability to read pro-

grams. Students involved in team programming had better programming scores while they spent less time than those students working individually.

Based on this study, it appears that the needs of industry (skill development in the area of team programming) can be met without detracting from the student's development of reading or writing abilities in traditional courses while team programming may detract from the student's learning of the language syntax.

BIBLIOGRAPHY

Baker, F. Terry. 1972. Chief programmer team management of production programming. IBM Systems Journal 11(1):56-73.

Baker, F. Terry, and Mills, Harlan D. 1973. Chief programmer teams. Datamation 19(12):58-61.

Barnlund, D. C. 1959. A comparative study of individual, majority, and group judgement. Journal of Abnormal and Social Psychology 58(1):55-60.

Black, Hubert P. 1969. The efficiency of the American College Testing Program and high school grades for predicting the achievement of Chesapeake College students. ERIC ED 029 626.

Bloom, Benjamin S. 1963. Testing cognitive ability and achievement. Pp. 379-397 in N. L. Gage, ed. Handbook of Research on Teaching. Rand McNally, Chicago.

Bohl, Marilyn. 1980. Information Processing. Science Research Associates, Inc., Chicago.

Bordon, R. B. 1976. Foreign language learning via the Lozanov method: pilot studies. Journal of Suggestive-Accelerative Learning and Teaching 1(1):3-15.

Borg, Walter R., and Gall, Meredith D. 1979. Educational Research. Longman, Inc., New York.

Brooks, Frederick P., Jr. 1975. The Mythical Man-Month. Addison-Wesley Publishing Co., Reading, Mass.

Campbell, Donald T., and Stanley, Julian C. 1963. Experimental and quasi-experimental designs of research on teaching. Pp. 171-276 in N. L. Gage, ed. Handbook of Research on Teaching. Rand McNally, Chicago.

Cheney, Paul H. 1977. Teaching computer programming in an environment where collaboration is required. AEDS Journal 11(3):1-5.

Clutterham, D. R. 1970. A method for evaluating student progress in undergraduate computer science by use of automated program sets. Final report. Office of Education, Bureau of Research, Washington, D. C. ERIC ED 051 665.

Dahl, O. J., Dijkstra, E. W., and Hoare, C. A. R. 1972. Structured Programming. Academic Press, Inc., London.

Davis, J. H. and Restle, F. 1963. The analysis of problems and prediction of group problem solving. Journal of Abnormal and Social Psychology 66(2):103-116.

Funches, De Lars. 1967. Correlations between secondary school transcript averages and grade point averages and between ACT scores and grade point averages of freshmen at Jackson State College. College and University 43(1):52-54.

Goldman, Morton. 1965. A comparison of individual and group performance for varying combinations of initial ability. Journal of Personality and Social Psychology 1(3):210-216.

Gruenberger, Fred. 1964. The teaching of computing. Rand Report P-2990.

Husband, Richard W. 1940. Cooperative versus solitary problem solving. Journal of Social Psychology 11(2):405-409.

Jackson, Durward P. 1980. Deskilling programming: management issues. Computerworld 14(46):49-54.

Judine, Sister M., I. H. M. 1965. A Guide for Evaluating Student Composition. National Council of Teachers of English, Champaign, Illinois.

Kenny, David A. 1975. A quasi-experimental approach to assessing treatment effects in a nonequivalent control group design. Psychological Bulletin 82(3):345-362.

Kernighan, Brian W., and Plauger, P. J. 1974. The Elements of Programming Style. McGraw-Hill Book Company, New York.

Khailany, Asad, and Saxon, Charles. 1978. Conducting project team classes in data processing. SIGCSE Bulletin 10(1):189-192.

Laughlin, Patrick R., and Johnson, Homer H. 1966. Group and individual performance on a complementary task as a function of initial ability level. Journal of Experimental Social Psychology 2(4):407-414.

Laughlin, Patrick R., McGlynn, Richard P., Anderson, Jon A., and Jacobson, Everett S. 1968. Concept attainment by individuals versus cooperative pairs as a function of memory, sex, and concept rule. Journal of Personality and Social Psychology 8(4):410-417.

Laughlin, Patrick R., Kerr, Norbert L., Davis, James H., Halff, Henry M., and Marciniak, Kenneth A. 1975. Group size, member ability, and social decision schemes on an intellective task. Journal of Personality and Social Psychology 31(3):522-535.

Lemos, Ronald S. 1977. A comparative study of the effectiveness of team interaction in COBOL programming language learning. Unpublished Ph.D. dissertation, UCLA.

Lemos, Ronald S. 1978. The cost-effectiveness of team debugging in teaching COBOL programming. SIGCSE Bulletin 10(1):193-196.

Lemos, Ronald S. 1979. An implementation of structured walk-throughs in teaching COBOL programming. Communications of the ACM 22(6):335-340.

Litecky, Charles R., and Davis, Gordon B. 1976. A study of errors, error-proneness and error diagnosis in COBOL. Communications of the ACM 19(1):33-37.

Mayer, Richard E. 1975. Different problem-solving competencies established in learning computer programming with and without meaningful models. Journal of Educational Psychology 67(6):725-734.

McGowan, Clement. 1975. Structured programming: a review of some practical concepts. Computer 8(6):25-30.

Miller, Nancy E., and Petersen, Charles G. 1980. A method for evaluating student written computer programs in an undergraduate computer science programming language course. SIGCSE Bulletin 12(4):9-17.

Munday, Leo. 1968. Correlations between ACT amd other predictors of academic success in college. College and University 44(1):67-76.

Pearl, Andrew Wilder. 1967. A study of the effects on students' achievement and attitudes when they work in academic teams of three members. Unpublished Ed.D. dissertation, Cornell University.

Perlmutter, H. V., and De Montmollin, G. 1952. Group learning of nonsense syllables. Journal of Abnormal and Social Psychology 47(4):762-769.

Philippakis, A. S. 1973. Programming language usage. Datamation 19(10):109-114.

Remigius, David. 1979. Declining ACT scores and grade inflation at Southeastern Louisiana University. ERIC ED 177 992.

Sawyer, Richard, and Maxey, E. James. 1979. The validity over time of college freshman grade prediction requirements. ERIC ED 185 896.

Schonberger, Richard J., and Franz, Lori. 1978. The required business computing course: peer-group learning with a managerial emphasis. AEDS Journal 11(3):73-83.

Shaw, Marjorie E. 1932. A comparison of individuals and small groups in the rational solution of complex problems. American Journal of Psychology 44(3):491-504.

Shaw, Marvin E. 1971. Groups Dynamics: The Psychology of Small Group Behavior. McGraw-Hill Book Co., New York.

Sippl, Charles J., and Sippl, Charles P. 1974. Computer Dictionary. Howard W. Sames & Co., Indianapolis.

Taylor, D. W., and Faust, W. L. 1952. Twenty questions: efficiency of problem solving as a function of the size of the group. Journal of Experimental Psychology 44(5):360-363.

Tuckman, J., and Lorge, I. 1962. Individual ability as a determinant of group superiority. Human Relations 15(1):45-51.

Ulloa, Miguel. 1980. Teaching and learning computer programming: a survey of student problems, teaching methods, and automated instructional tools. SIGCSE Bulletin 12(2):48-64.

Van Tassel, Dennie. 1978. Program Style, Design, Efficient, Debugging, and Testing. Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Weinberg, Gerald M. 1971. The Psychology of Computer Programming. Van Nostrand Reinhold, New York.

Weinberg, Gerald M., and Schulman, Edward L. 1974. Goals and performance in computer programming. Human Factors 16(1):70-77.

Weinberg, Gerald M., Wright, Stephen E., Kauffman, Richard, and Goetz, Martin A. 1977. High Level COBOL Programming. Winthrop Publishers, Inc., Cambridge, Mass.

Youngs, Edward A. 1974. Human errors in programming. International Journal of Man-Machine Studies 6(3):361-376.

Yuker, H. E. 1955. Group atmosphere and memory. Journal of Abnormal and Social Psychology 51(1):17-25.

Ziller, R. C. 1957. Group size: A determinant of the quality and stability of group decisions. Sociometry 20(2):165-173.

ACKNOWLEDGEMENTS

The author would like to thank the following people for making this dissertation possible. First, the investigator's graduate committee for the constructive criticism given and interest shown during the preparation of this dissertation. Second, the students enrolled in Introduction to COBCL Programming at the University of Wisconsin - La Crosse during the Spring and Fall semesters of 1980 who participated in the experiment. Third, the instructors at the University of Wisconsin - La Crosse for their support and cooperation in conducting the research. Finally, to my husband, Chuck, for his emotional support and his help in making this dissertation comprehensible.

APPENDIX A.

BIOGRAPHICAL QUESTIONNAIRE

Name_____
Last First Middle Initial

Major_____ Section_____

Year in School(circle one): Fr So Jr Sr Graduate Special

Reason for taking the course:

_____Elective in Basic Studies

_____Required in the Major

_____Elective in Major, not required

Sex(circle one): Male Female

Age_____

Specify which Computer Science courses you have taken prior to this semester by circling the semester and filling in the year you took the course.

110	FALL SPG SUM 19___		361	FALL SPG SUM 19___	
124	FALL SPG SUM 19___		365	FALL SPG SUM 19___	
221	FALL SPG SUM 19___		370	FALL SPG SUM 19___	
222	FALL SPG SUM 19___		441	FALL SPG SUM 19___	
323	FALL SPG SUM 19___		442	FALL SPG SUM 19___	
331	FALL SPG SUM 19___		451	FALL SPG SUM 19___	
340	FALL SPG SUM 19___		452	FALL SPG SUM 19___	
351	FALL SPG SUM 19___		453	FALL SPG SUM 19___	
360	FALL SPG SUM 19___		470	FALL SPG SUM 19___	

Coding of Biographical Items:

Major

1-Computer Science
2-Business
3-Other

Year in college

1-freshman
2-sophomore
3-junior
4-senior
5-graduate and special

Reason for taking the course

1-elective in basic studies
2-required in the major
3-elective in major, not required

Sex

1-female
2-male

Computer Science courses taken

0-neither FORTRAN nor PASCAL
1-FORTRAN (prior to Fall 79)
2-PASCAL (Fall 79 and after)

APPENDIX B.

ESTIMATE OF TIME SPENT FORM

ON PROGRAM (CIRCLE ONE) 1 2 3 4 5 6

Name_____
Last First Middle Initial

Section_____

Please give time estimates to the nearest tenth of an hour.

Estimate of time for design_____

Estimate of time for coding_____

Estimate of time for testing and debugging_____

APPENDIX C.

EVALUATION OF TEAM MEMBER FORM

Name of team member to be evaluated_____
Last
First

for program (circle one): 1 2 3 4 5 6

Please rate the above team member on the following items using a scale of 1 to 5. Please do not sign your name.

1=POOR 2=FAIR 3=SATISFACTORY 4=GOOD 5=EXCELLENT

1. Effort in designing program _____
2. Effort in coding program _____
3. Effort in testing and debugging program _____
4. Overall participation of this person as a team member _____

APPENDIX D.

COMPUTER PROGRAM EVALUATION FORM

I. Minimum requirements to be met (80%)	
A. Flowchart (10%)	
B. Program style and clarity (35%)	
1. Comments (10%)	
a. Author at beginning (2%)	
b. Remarks at beginning (3%)	
c. Comments throughout program (5%)	
2. Meaningful data names (10%)	
3. Structured listing (15%)	
a. Indentation to show structure (10%)	
b. modularity of design (5%)	
C. Output clarity (35%)	
1. Correct output (partial-% of correct) (20%)	
2. Appropriate labeling (10%)	
3. Correct termination of program (5%)	
II. Refinements above minimum (20%)	
A. Program length (5%)	
B. Output embellishments (5%)	
C. Exemplary program style and clarity (10%)	
TOTAL	

APPENDIX E.

PRETEST

MULTIPLE CHOICE. Choose the one best answer.

1. Which of the following rules correctly applies to the PICTURE (PIC) clauses?

   a) maximum number of characters in a PIC string is 18
   b) maximum number of digits stored in a PIC is 18
   c) edited items are considered numeric
   d) the zero insertion may be used with alpha PIC's only
   e) edited items may be used in computations

2. In subscripting a table, the clause that must be used is

   a) REDEFINES clause
   b) OCCURS clause
   c) USAGE clause
   d) INDEXED BY clause
   e) ASCENDING/DESCENDING KEY clause

3. The INDEXED BY clause

   a) specifies that a data name defined elsewhere in the DATA DIVISION to be an index name.
   b) defines an index name to be used in any way the programmer sees fit.
   c) is always necessary with an OCCURS clause.
   d) establishes a variable to be an index name with no other definition permitted.
   e) is necessary with a SEARCH command but not the SEARCH ALL command.

4. The purpose of the continuation indicator (hyphen in column 7) is to

   a) continue a non-numeric literal on the next line.
   b) provide comments to the program.
   c) complete a statement on a succeeding line.
   d) continue a literal and complete a statement on the next line.
   e) indicate the end of the paragraph.

5. If data are stored in a binary format, the following USAGE clause is used:

   a) USAGE IS DISPLAY.
   b) USAGE IS CCMPUTATICNAL.
   c) USAGE IS COMPUTATIONAL-1.
   d) USAGE IS CCMPUTATICNAL-2.
   e) USAGE IS CCMPUTATICNAL-3.

6. If data are stored in a packed decimal format, the following USAGE clause is used:

   a) USAGE IS DISPLAY.
   b) USAGE IS CCMPUTATIONAL.
   c) USAGE IS CCMPUTATICNAL-1.
   d) USAGE IS CCMPUTATIONAL-2.
   e) USAGE IS CCMPUTATICNAL-3.

7. Which one of the fcllowing is a valid COBOL data name?

   a) 2ND-PARA
   b) INPUT-OUTPUT
   c) RECORD-INPUT
   d) ALPHA/NAME
   e) DATA

8. Which one of the following is a valid COBOL numeric literal?

   a) $123.45
   b) 1,234.00
   c) 12.
   d) 1.2
   e) .05-

9. Which one of the fcllowing statements is syntactically correct?

   a) ADD A, B, C, TO D GIVING F.
   b) DIVIDE R BY S.
   c) DIVIDE A INTO 6.
   d) SUBTRACT A, B FROM C.
   e) MULTIPLY A BY C GIVING D, E.

10. The one rule that applies to the assignment of names in COBOL is:

a) names may be formed from any combination of alphanumeric characters.
b) no hyphen may appear within a name.
c) names may range up to 30 characters in length.
d) only alphabetic characters may be used.
e) only alphabetic characters and numeric digits may be used.

11. Numeric literals

a) may contain up to 30 digits.
b) are enclosed in quotations.
c) may contain a sign in the rightmost or leftmost position.
d) may contain up to 15 digits.
e) may be edited.

12. Which of the following statements is false concerning editing pictures?

a) there must be either an actual decimal or an assemed decimal point.
b) there must be at least one digit position character.
c) there may be only one type of floating string characters.
d) there cannot be a mixture of floating strings or replacement characters.
e) there may be as many as 30 characters in the edited picture clause.

13. The ENVIRONMENT DIVISION contains which one of the following sections?

a) FILE SECTION
b) WORKING-STORAGE SECTION
c) INPUT-OUTPUT SECTION
d) LINKAGE SECTION
e) DATA SECTION

14. One of the follcwing is not a class condition that may be used in COBCL

a) POSITIVE
b) NEGATIVE
c) ALPHABETIC
d) NUMERIC
e) ALPHANUMERIC

15. The SPECIAL NAMES paragraph is:

a) used to equate mnemonic names with actual machine devices.
b) written in the INPUT-OUTPUT SECTION.
c) used to assign a name to a certain system function.
d) used to define a function.
e) written in the DATA DIVISION.

16. Which of the fcllowing editing symbcls is <u>not</u> considered an insertion character?

a) B
b) $
c) 0
d) +
e) Z

Match the terms with the descriptions in questions 17-22 concerning the SEARCH statement. The terms may be used more than one.

a - SET statement
b - ASCENDING/DESCENDING KEY clause
c - INDEXED BY clause
d - SEARCH statement
e - SEARCH ALL statement

17. is necessary only with the SEARCH ALL statement.

18. is necessary for both forms of the SEARCH statement.

19. specifies that a sequential search is to be done.

20. is used to initialize, increment, and decrement index names.

21. is necessary cnly with the sequential search.

22. may equate values of data names or constants with index names.

Match the terms with the descriptions in questions 23-24 concerning the SORT statement. The terms may be used more than once.

a - INPUT PROCEDURE clause
b - OUTPUT PROCEDURE clause
c - USING clause
d - GIVING clause
e - ON ASCENDING/DESCENDING KEY clause

23. requires the use of the RETURN statement.

24. requires the use of the RELEASE statement.

25. One of the rules governing arithmetic statements is:

a) all literals used must be non-numeric.
b) the GIVING option may be used with the COMPUTE verb.
c) editing symbols may not be used on data names following GIVING.
d) all literals used must be numeric.
e) the GIVING may be used with the ADD only.

Assume the following pictures for questions 26-29. For each of the statements and values given, chose the answer that is the result of the calculation <u>as stored in the receiving field</u>. b is a blank, 0 is a zero).

```
02  Q  PICTURE 99.
02  R  PICTURE 9V9.
02  S  PICTURE 99V9.
02  T  PICTURE 99V99
02  W  PICTUREV 99.
```

26. Given Q = 7 R = 0.3 S = 0.5
ADD Q, R GIVING S.

a) 7.3
b) 07v8
c) 07v3
d) 7v3
e) 7v8

27. Given Q = 16 R = 6.2
SUBTRACT R FRCM Q.

a) 9.8
b) 09
c) 10
d) 9v8
e) 09v0

28. Given Q = 7 R = 0.3
MULTIPLY Q BY R.

a) 2.1
b) 02
c) 2
d) 2v1
e) 02v1

29. Given W = 0.75
DIVIDE 2 INTO W ROUNDED.

a) v37
b) v375
c) v38
d) v75
e) 3v75

30. Assume the following PICTURES:

```
02  H  PICTURE 999V9.
02  L  PICTURE 999V99  VALUE 132.45.
```

choose the answer that is the result of the following MOVE statement as stored in the receiving field.

```
MOVE L TO H.
```

a) 132v4
b) 132v5
c) 132v45
d) 132.5
e) 324v5

31. Given the following:

SOURCE PICTURE = S9(4)V9
SOURCE DATA = 0125v6
RECEIVING PICTURE = ++++9.9

After moving from the source to the receiving field, the receiving field will contain:

a) ++++5.6
b) +b125.6
c) -b125.6
d) b-125.6
e) b+125.6

How may times will the procedure named ROUTINE-X be executed by the PERFORM statements in questions 32-34?

32.

```
PERFORM RCUTINE-X
    VARYING SUB FRCM 1 BY 1
    UNTIL SUB IS GREATER THAN 10.
```

a) 1
b) 0
c) 11
d) 10
e) 9

33.

```
PERFORM ROUTINE-X
    VARYING SUB FROM 1 BY 1
    UNTIL SUB IS LESS THAN 10.
```

a) 9
b) 10
c) 11
d) 0
e) 1

34.
```
PERFORM RCUTINE-X
    VARYING SUB FROM 1 BY 1
        UNTIL SUB IS GREATER THAN 3
    AFTER POSITION FRCM 2 BY 2
        UNTIL POSITION IS GREATER THAN 6.
```

a) 6
b) 8
c) 7
d) 3
e) 9

35. Which one of the following conditions has a different evaluation result?

a) A=1 OR A=2 AND B=4
b) A=1 OR (A=2 AND B=4)
c) A=1 OR =2 AND B=4
d) (A=1 OR =2) AND B=4
e) A=1 OR 2 AND B=4

36. The value +123 stored in DISPLAY form would be stored in HEX representation as:

a) F1 F2 F3
b) F1 F2 C3
c) 00 01 23
d) F1 F2 D3
e) F+ F1 D2 F3

37. An OPEN statement:

a) must be executed prior to any other input or output instruction for the file.
b) can name only one file.
c) makes input records available for processing.
d) can appear only once in the PROCEDURE DIVISION.
e) can open either input files or output files in one statement but not both.

38. The RELEASE statement:

a) can appear in the OUTPUT PROCEDURE.
b) need not be included in the INPUT PROCEDURE.
c) causes records to be transferred to the sort file.
d) causes records to be input from the sort file.
e) must be used in the OUTPUT PROCEDURE.

Questions 39-41 pertain to the following program.

```
WORKING-STORAGE SECTION.
77  POSITION        PICTURE 99.
77  SUB             PICTURE 99    VALUE 3.
77  CALC            PICTURE 9(4)  VALUE 5.
01  TABLE.
    03  INFO.
        05  FILLER PICTURE 99    VALUE 10.
        05  FILLER PICTURE 99    VALUE 20.
        05  FILLER PICTURE 99    VALUE 30.
        05  FILLER PICTURE 99    VALUE 40.
        05  FILLER PICTURE 99    VALUE 50.
    03  ARRAY REDEFINES INFO.
        05  NUMBER OCCURS 5 TIMES PICTURE 99.
PROCEDURE DIVISION.
MAINLINE.
    PERFORM CALCULATION
        VARYING SUB FROM 1 BY 1
        UNTIL SUB IS GREATER THAN 3.
    STOP RUN.
CALCULATION.
   ADD 2, SUB GIVING POSITION.
    MULTIPLY NUMBER (SUB) BY NUMBER (POSITION)
        GIVING CALC.
```

39. The first value output is

    a) 0300
    b) 0005
    c) 1500
    d) 0008
    e) 0800

40. The second value output is

    a) 0300
    b) 0005
    c) 1500
    d) 0008
    e) 0800

41. The third value output is:

    a) 0300
    b) 0005
    c) 1500
    d) 0008
    e) 0800

42. If the SIZE ERROR clause is used, and a size error condition arises,

a) the result is unpredictable.
b) the size of the receiving field is enlarged to hold the value.
c) the error exists in the receiving field since leading digits are truncated.
d) the value of the receiving field remains unchanged.
e) the program stops executing.

43. Which of the following is false concerning a level 88 entry?

a) assigns a name to a specific value that a data item may assume.
b) may be used in the FILE SECTION and the WORKING-STORAGE SECTION.
c) may be used with 77 levels.
d) does not have a PICTURE clause.
e) assigns an initial value to an elementary item.

44. Which of the following is true concerning the JUSTIFIED clause?

a) the justified clause causes data to be moved into the receiving field from right to left.
b) the justified clause causes data to be moved into the receiving field from left to right.
c) the justified clause may be used with any picture clause.
d) the justified clause is used with numeric pictures to store the values against the right hand side.
e) the justified clause is treated by the compiler as comments.

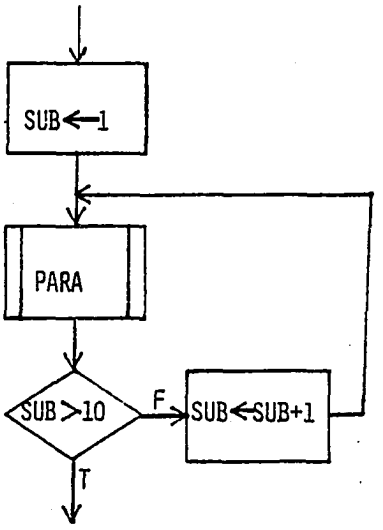
45. The RETURN statement:

a) causes one record to be transferred to the sorting operation.
b) can appear only in the INPUT PROCEDURE.
c) can appear in either the INPUT PROCEDURE or the OUTPUT PROCEDURE.
d) retrieves a record from the sort file.
e) can be used with the USING clause.

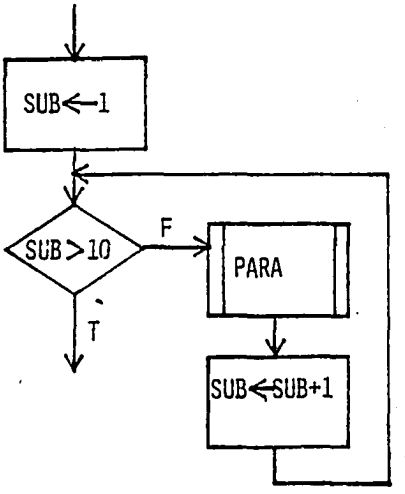
46. Which of the following flowcharts represent the order of steps taken when executing the following statement:

```
PERFORM PARA
   VARYING SUB FROM 1 BY 1
   UNTIL SUB GREATER THAN 10.
```

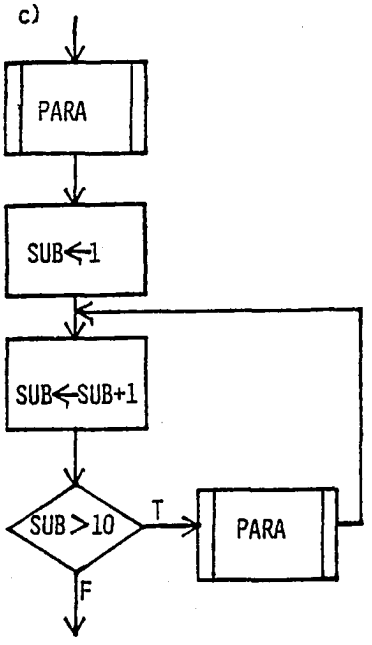
A)



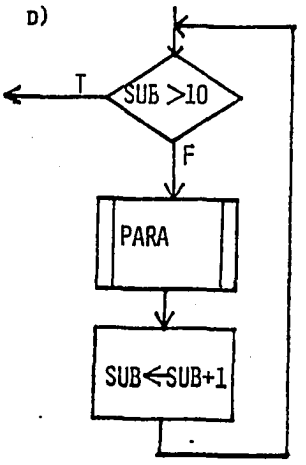
B)



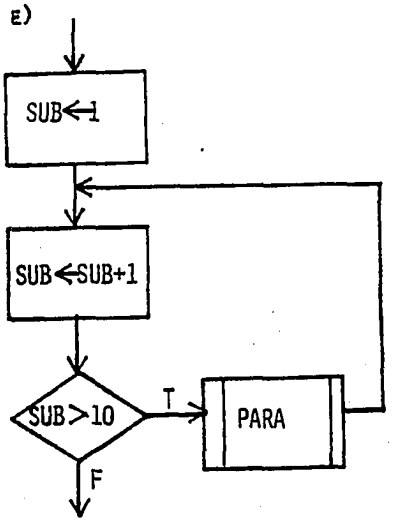
C)



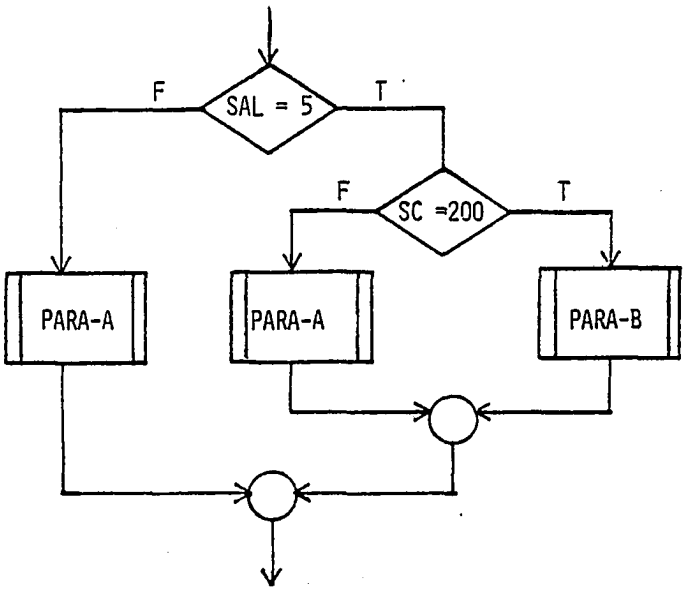
D)



E)



Question 47 pertains to the following flowchart.



47. Which of the following statements represent the flowchart?

a)
```
IF SC = 200
 THEN PERFORM PARA-B
 ELSE PERFORM PARA-A.
```
b)
```
IF SC = 200
 THEN PERFORM PARA-B
 ELSE IF SAL = 5
    THEN PERFORM PARA-A
    ELSE PERFORM PARA-A.
```
c)
```
IF SAL NOT = 5 AND SC = 200
 THEN PERFORM PARA-A
 ELSE PERFORM PARA-B.
```
d)
```
IF NOT SAL = 5 OR NOT SC = 200
 THEN PERFORM PARA-A
 ELSE PERFORM PARA-B.
```
e)
```
IF NOT SAL = 5 OR SC = 200
 THEN PERFORM PARA-A
 ELSE PERFORM PARA-B.
```

Match the terms with the descriptions in questions 48-50 concerning the SORT verb.

a - INPUT PROCEDURE clause
b - OUTPUT PROCEDURE clause
c - USING clause
d - GIVING clause
e - ON ASCENDING/DESCENDING KEY clause

48. allows the programmer to pass a record one at a time to the sort file.

49. file will be automatically opened, read from, and closed.

50. allows the programmer to receive a record one at a time from the sort file.

APPENDIX F.
COMPUTER GRADING FORM STATISTICS

TABLE F1
Raw Scores and Ranks

PROFESSOR 1		PROFESSCR 2		PROFESSOR 3	
85	2	88	3	79	3.5
82	4	85	5	79	3.5
70	9	80	7	66	9
86	1	89	1.5	78	5
82	4	82	6	82	1
74	7	86	4	69	8
71	8	77	9	70	7
78	6	78	8	73	6
45	10	54	10	41	10
82	4	89	1.5	81	2
AVERAGE	75.5		80.8		71.8
STANDARD DEVIATION	12.1		10.4		12.1

TABLE F2
Analysis of Variance of Raw Scores

SOURCE OF VARIATION	SS	DF	MEAN SQUARE	F	PROBABILITY
BETWEEN PEOPLE	3471.63	9	385.74		
WITHIN PEOPLE	553.33	20	27.67		
BETWEEN MEASURES	409.27	2	204.63	25.57	0.00001
RESIDUAL	144.07	18	8.00		
NONADDITIVITY	17.93	1	17.93	2.42	0.14
BALANCE	126.12	17	7.42		
TOTAL	4024.97	29	138.79		

TABLE F3
Correlation Matrix for Program Scores
Ranked
Raw

	PROFESSOR 1	PROFESSOR 2	PROFESSOR 3
PROFESSOR 1	1.000		
	1.000		
PROFESSOR 2	.812**	1.000	
	.943**	1.000	
PROFESSOR 3	.794**	.585**	1.000
	.979**	.915**	1.000
RELIABILITY COEFFICIENT (ALPHA) =		.890	
		.979	

** $p < 0.01$.

APPENDIX G.

OUTLINE FOR CPTS 222

Weekly Periods	Topic	Program
1	Introduction Procedure Division	
2	Data Division, Data names Record descriptions Levels 01-46, Picture clauses	Program #1 (Simple input and output)
3	Working storage section, Move Add,subtract,multiply,divide	
4	Exam #1 Review exam Display statement	Program #2 (Arithmetic calculations
5	Structured flowcharts If, Relational conditions	
6	Value clause, Class conditions Logical operators Write from statement	Program #3 (Intermediate totals)
7	Edited pictures, 77, 88 levels Write advancing statement	
8	Midterm exam #2 Review exam, Discuss program	Program #4 (Arrays)
9	Subscripts Arrays, Occurs Perform varying, Redefines	
10	Search verb Indexed by clause Set statement Ascending key clause	Program #5 (Search verb)
11	Identification division Nested if statements Justified clause Implied subjects	
12	Sync clause, Examine statement Move corr statement	
13	Exam #3 Review exam, Discuss program	Program #6 (Sort verb)
14	Sort statement	
15	Compare statement, Subroutines Sequential file processing	
16	Blocking clause Renames clause Go to depending on statement Indexed sequential files Direct file organization	
17	Final exam	

APPENDIX H.

HUMAN SUBJECTS FORM

INFORMATION ON THE USE OF HUMAN SUBJECTS IN RESEARCH
IOWA STATE UNIVERSITY
(Please follow the accompanying instructions for completing this form.)
114

1. Title of project (please type): The effectiveness of team programming on student achievement in Introduction to COBOL Programming at University of Wisconsin-La Crosse.

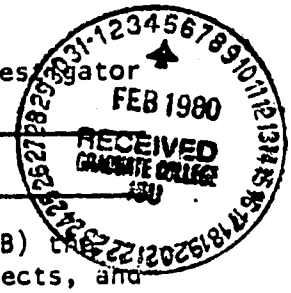
2. I agree to provide the proper surveillance of this project to insure that the rights and welfare of the human subjects are properly protected. Additions to or changes in procedures affecting the subjects after the project has been approved will be submitted to the committee for review.

Nancy E. Miller — 12-27-79 — Nancy E. Miller
Typed Named of Principal Investigator — Date — Signature of Principal Investigator

2131 S. 29th Street
La Crosse, Wisconsin 54601 — 608-788-8046
Campus Address — Campus Telephone

3. Signatures of others (if any) — Date — Relationship to Principal Investigator

[illegible] — Major Professor
[illegible] Comp Center



4. ATTACH an additional page(s) (A) describing your proposed research and (B) the subjects to be used, (C) indicating any risks or discomforts to the subjects, and (D) covering any topics checked below. CHECK all boxes applicable.

- ☐ Medical clearance necessary before subjects can participate
- ☐ Samples (blood, tissue, etc.) from subjects
- ☐ Administration of substances (foods, drugs, etc.) to subjects
- ☐ Physical exercise or conditioning for subjects
- ☐ Deception of subjects
- ☐ Subjects under 14 years of age and(or) ☐ Subjects 14-17 years of age
- ☐ Subjects in Institutions
- ☒ Research must be approved by another institution or agency

5. ATTACH an example of the material to be used to obtain informed consent and CHECK which type will be used.

- ☐ Signed informed consent will be obtained.
- ☐ Modified informed consent will be obtained.

6. Anticipated date on which subjects will be first contacted: Month 2 Day 1 Year 80

Anticipated date for last contact with subjects: Month 5 Day 30 Year 80

7. If Applicable: Anticipated date on which audio or visual tapes will be erased and(or) identifiers will be removed from completed survey instruments:
Month ___ Day ___ Year ___

8. Signature of Head or Chairperson — Date 1/5/80 — Department or Administrative Unit: Educational Studies

9. Decision of the University Committee on the Use of Human Subjects in Research:

☒ Project Approved ☐ Project not approved ☐ No action required

George G. Karas — 2/7/80 — G G Karas
Name of Committee Chairperson — Date — Signature of Committee Chairperson

Revised 6/79